UNIVERSIDADE DO VALE DO TAQUARI - UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA ELÉTRICA

DETECÇÃO E CLASSIFICAÇÃO DE FALTAS EM LINHAS DE TRANSMISSÃO UTILIZANDO REDES NEURAIS ARTIFICIAIS

Estevan Luiz Junges

Lajeado, junho de 2018

Estevan Luiz Junges

DETECÇÃO E CLASSIFICAÇÃO DE FALTAS EM LINHAS DE TRANSMISSÃO UTILIZANDO REDES NEURAIS ARTIFICIAIS

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas e aprovado em sua forma final pelo orientador e pela banca examinadora.

Prof. Dr. Ederson P. Madruga – orientador
Universidade do Vale do Taquari

Prof. Ms. Alexandre Stürmer Wolf
Universidade do Vale do Taquari

Prof. Ms. Anderson Antônio Giacomolli
Universidade do Vale do Taquari

Lajeado, junho de 2018

Estevan Luiz Junges

DETECÇÃO E CLASSIFICAÇÃO DE FALTAS EM LINHAS DE TRANSMISSÃO UTILIZANDO REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas Universidade do Vale do Taquari - Univates, como parte dos requisitos para a obtenção do título de bacharel em Engenharia Elétrica

ORIENTADOR: Prof. Dr. Ederson P. Madruga

Lajeado, junho de 2018

AGRADECIMENTOS

Agradeço primeiramente ao meus pais, Ledir e Elton, que sempre me apoiaram, de todas as maneiras possíveis, na minha caminhada. O seu empenho e dedicação são o que me levam, e o que me inspiram, a seguir em frente.

Agradeço a amada companheira Inauã que esteve comigo não só durante o árduo período de desenvolvimento do trabalho de conclusão, mas durante grande parte de minha vida, sempre apoiadora de meu trabalho de das minhas escolhas.

Agradeço ao apoio do orientador Professor Ederson Madruga, que acreditou no meu potencial para a realização do trabalho. Também agradeço a instituição, a todo o corpo docente e a coordenação do curso, que sempre estiveram dispostos a ajudar quando foi necessário.

Por fim, agradeço a todos amigos, colegas e familiares que, de alguma forma, direta ou indireta, me auxiliaram no desenvolvimento do trabalho.

RESUMO

O presente trabalho apresenta uma proposta de um sistema de detecção e classificação de faltas em linhas de transmissão, baseado em redes neurais artificiais. O banco de dados para o treinamento e validação da rede neural foi gerado através do *software* de análise de transientes ATP, com o auxílio de um programa computacional desenvolvido para gerar o arquivo de circuito base da simulação de cada caso. A implementação da rede neural foi feita na linguagem Python, com a utilização das bibliotecas Keras, Tensorflow e Scikit-Learn especificas para aplicações de *machine learning*. A rede neural desenvolvida possui 6 entradas, que corresponde a corrente e a tensão de cada fase da linha de transmissão, e 11 saídas, que correspondem a cada um dos estados de operação do sistema. Os resultados obtidos mostram que a utilização de redes neurais artificiais é viável para a detecção e classificação de faltas.

Palavras-chave: Transmissão, Detecção de Faltas, Redes Neurais Artificiais

ABSTRACT

This document proposes a transmission line fault detection and classification system, based on artificial neural networks. The dataset used for training and validating the neural network was generated with the transient analysis software ATP, aided by a computational program developed to create the base circuit file for each simulation case. The neural network implementation was made with the Python programing language and using the libraries Keras, Tensorflow and Scikit-Learn, that are specific to machine learning applications. The proposed neural network has 6 inputs that correspond to the voltage and current of each transmission line phase and 11 outputs that corresponds to each one of the system estates. The results show that artificial neural networks are a viable option for the problem of detection and classification of electrical faults in transmission lines.

Keywords: Transmission, Fault Detection, Artificial Neural Networks

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABNT:	Associação Brasileira de Normas Técnicas
API:	Application Programing Interface
ATP:	Alternate Transients Program
CEPEL:	Centro de Pesquisa de Energia Elétrica
CETEC:	Centro de Ciências Exatas e Tecnológicas
CPU:	Central Processing Units
CSV:	Comma-Separated Values
DLG:	Double line-to-ground
EMTP:	ElectroMagnetic Transient Program
FAI:	Falta de Alta Impedância
GPU:	Graphics Processing Unit
IDE:	Integrated Development Environment
IEEE:	Institute of Electrical and Electronics Engineers
JVM:	Java Virtual Machine
LL:	Line-to-line
LT:	Linha de Transmissão
ReLU:	Rectified Linear Units
RNA:	Rede Neural Artificial
SEP:	Sistema Elétrico de Potência
SGBD:	Sistema Gerenciador de Banco de Dados
SLG:	Single line-to-ground
SQL:	Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

A geração e a utilização de energia elétrica, nos dias de hoje, são consideradas imprescindíveis para o funcionamento da sociedade. Instituições, sejam elas públicas ou privadas, em diversos setores, desde a produção agrária até à indústria, dependem de um fornecimento ininterrupto de eletricidade. Esse fornecimento deve ser de qualidade, seguro e com baixo custo. Logo, pesquisas nessa área são de grande interesse (PRASAD; EDWARD, 2017).

A infraestrutura responsável por gerar a energia elétrica e entregá-la aos consumidores finais é conhecido como Sistema Elétrico de Potência e é normalmente dividido em Geração, Transmissão e Distribuição (DAS, 2006). Dentre as 3 subdivisões, os componentes mais propensos a faltas são as linhas de transmissão devido à sua grande susceptibilidade a descargas atmosféricas. Logo, a proteção adequada desses equipamentos é vital para o funcionamento da cadeia de fornecimento de energia elétrica (OLESKOVICZ, 2001).

O sistema de proteção adotado deve ser capaz de detectar rapidamente a falta, além de elencar quais fases estão presentes e ainda determinar a sua localização. Essas informações são cruciais para agilizar os reparos e, consequentemente, reduzir o tempo de indisponibilidade do equipamento (OLIVEIRA, 2005). Para isso é importante o desenvolvimento de sistemas de proteção de qualidade.

Embora uma infinidade de eventos possa causar faltas em uma rede de transmissão, os sistemas de proteção devem ser capazes de discriminá-las em quatro tipos, segundo Oleskovicz (2001): um condutor à terra (fase-terra), dois condutores à terra (fase-fase-terra), dois condutores (fase-fase) e três condutores (fase-fase-fase).

Além dos diversos tipos de faltas, outro fator que dificulta a detecção, classificação e localização de falhas, principalmente no caso de falta fase-terra, é a existência de uma

impedância de contato. Dependendo da magnitude dessa impedância a falta pode passar despercebida pelos sistemas de proteção convencional. Isso leva a busca de novas formas de desenvolvimento de proteções de linhas de transmissão (BAQUI *et al.*, 2011).

Uma alternativa aos sistemas de proteção convencionais é a utilização de Redes Neurais Artificiais (RNA). O uso de RNA tem gerado ótimos resultados nas mais diversas áreas dos sistemas de potência. Dentre elas: previsão de demanda e geração, controladores e estabilizadores de tensão e frequência, despacho econômico, processamento de alarmes e diagnósticos de falta (VANKAYALA; RAO, 1993; MADAN; BOLLINGER, 1997).

As RNA são capazes de reconhecer padrões através de exemplos, por isso elas não necessitam de conhecimento específico sobre o problema abordado e não carregam a complexidade matemática dos sistemas convencionais. Além disso, elas são capazes de abordar problemas sobre os quais ainda não existe conhecimento específico, normalmente problemas complexos e não lineares (KALOGIROU, 2000). As RNA conseguem lidar bem com bases de dados incompletas, são robustas, tolerantes a falhas e utilizam menos processamento do que os sistemas convencionais, tornando mais fácil a utilização em aplicações online (HAQUE; KASHTIBAN, 2005).

A partir do que foi apresentado sobre a importância da proteção de linhas de transmissão e das evidências de sucesso da utilização de RNAs em sistemas de potência, o presente trabalho propõe a implementação de um sistema de detecção e classificação de faltas em linhas de transmissão baseado em redes neurais artificiais. É esperado que o sistema seja capaz de oferecer uma proteção adicional à estrutura existente.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um sistema de detecção e classificação de faltas em linhas de transmissão utilizando redes neurais artificiais. Esse sistema trabalhará em paralelo com os mecanismos de proteção já existentes e tem como objetivo proporcionar uma maior segurança para a operação do equipamento.

1.1.2 Objetivos Específicos

Os objetivos específicos do presente trabalho são:

1) Desenvolver uma solução que utilize somente as medidas de tensão e correntes trifásicas;
2) Utilizar tecnologias que facilitem o desenvolvimento, a experimentação e a manutenção do *software* desenvolvido;
3) Alcançar uma precisão de maior que 95% na detecção e classificação das falhas;
4) Criar uma base de dados que possa ser utilizada em outras pesquisas.

1.2 Estrutura e organização do trabalho

No capítulo 2 é apresentada uma breve revisão da literatura sobre faltas em linhas de transmissão. São apresentados os tipos de curtos-circuitos, como eles ocorrem e sua modelagem.

O referencial teórico sobre redes neurais artificiais é apresentado no capitulo 3. O capítulo inicia com uma visão geral do conceito de redes neurais e, posteriormente, é apresentado o neurônio biológico, o modelo de neurônio artificial, as arquiteturas de rede, os paradigmas de aprendizagem. Por fim, é detalhado o algoritmo de aprendizagem *Backpropagation*.

No capítulo 4 é apresentado o desenvolvimento dos procedimentos metodológicos de forma detalhada. Os resultados obtidos são apresentados no capítulo 5. A conclusão do trabalho é apresentada no capítulo 6.

2 FALTAS EM LINHAS DE TRANSMISSÃO

Em um Sistema Elétrico de Potência (SEP) a Transmissão é a subdivisão responsável por transportar a energia geradas pelas usinas para os centros de distribuição e consumo, com o mínimo de perdas. As linhas de transmissão (LT) são os principais componentes da Transmissão, e são compostas por torres, condutores, isoladores e para-raios (PINTO, 2014).

Por serem de grande porte e operarem em elevadas tensões, as LT são os componentes do SEP mais susceptíveis a defeitos. Por conseguinte, o foco dos sistemas de proteção é isolar as faltas no menor tempo possível para que a integridade da linha e a estabilidade do sistema não sejam ameaçadas (OLIVEIRA, 2005).

2.1 Tipos de curtos-circuitos

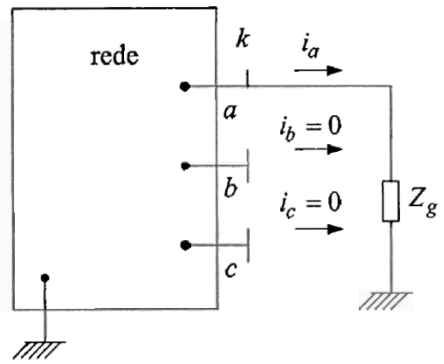
Nessa seção são detalhados os principais tipos de faltas que ocorrem nas linhas de transmissão.

2.1.1 Curto-circuito Fase-Terra

A Figura 1 apresenta o modelo elétrico de um curto-circuito fase terra, também chamado de *single line-to-ground* (SLG) (BERGEN; VITTAL, 2000).

O curto-circuito fase-terra ocorre quando uma das fases da linha entra em contato com o solo. Diversos fatores podem ocasionar essa situação, como fortes ventanias, acúmulo de neve ou gelo sobre o condutor e até mesmo a queda de uma árvore. Esse tipo de curto-circuito é o que ocorre com maior frequência e equivale a 70% do total de faltas em linhas de transmissão (FAULKENBERRY; COFFER, 1996).

Figura 1 - Curto-circuito Fase-Terra



Fonte: Zanetta, 2005, p. 183

A equação que descreve o curto-circuito fase-terra, de acordo com Zanetta (2005), está apresentada na Equação 1.

$$i_a = \frac{3e_1}{z_0 + 2z_1} \quad (1)$$

Onde:

i_a = corrente de curto-circuito fase-terra
e_1 = tensão da rede
z_0 = impedância de sequência zero
z_1 = impedância de sequência positiva

2.1.2 Curto-circuito Fase-Fase

O curto-circuito fase-fase, também chamado de *line-to-line* (LL) está representado na Figura 2 (BERGEN; VITTAL, 2000).

As falhas fase-fase ocorrem quando dois condutores de uma linha, de fases diferentes, entram em contato. Normalmente causada por fortes ventanias, que empurram um condutor para perto de outro, ou pelo rompimento e queda de um condutor sobre o outro, esse tipo de falta equivale a 15% dos casos (FAULKENBERRY; COFFER, 1996).

De acordo com Zanetta (2005), a equação que descreve o curto-circuito fase-fase está apresentada na Equação 2.

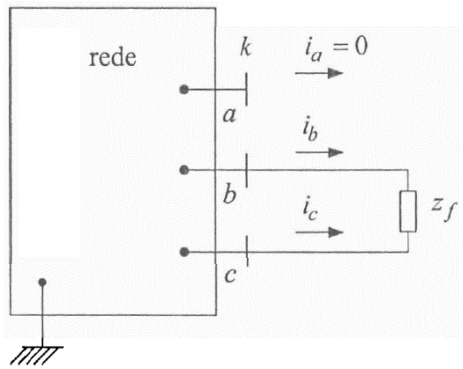
$$i_{2\phi} = \frac{\sqrt{3}}{2} i_{3\phi} \quad (2)$$

Onde:

$i_{2\phi}$ = tensão de curto-circuito fase-fase

$i_{3\phi}$ = tensão de curto-circuito trifásico

Figura 2 - Curto-circuito Fase-Fase

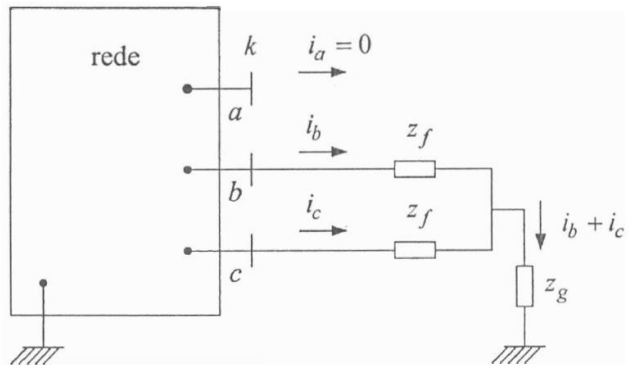


Fonte: Zanetta, 2005, p. 188.

2.1.3 Curto-circuito Fase-Fase-Terra

O curto-circuito fase-fase-terra, também chamado de *double line-to-ground* (DLG), está representado na Figura 3 (BERGEN; VITTAL, 2000).

Figura 3 - Curto-circuito Fase-Fase-Terra



Fonte: Zanetta, 2005, p. 192.

Os curtos-circuitos fase-fase-terra normalmente tem a mesma causa que curtos-circuitos fase-terra, porém, com o agravante de envolver duas linhas de transmissão. Como não são tão

frequentes, esse tipo de curto-circuito equivale somente a 10% dos casos (FAULKENBERRY; COFFER, 1996).

A Equação 3 descreve o curto-circuito fase-fase, segundo Zanetta (2005).

$$i_b = \left(z_2 e^{-j150^{\circ}} - j z_0\right) \frac{\sqrt{3}}{2z_0 + z_1} \frac{e_1}{z_1} \tag{3}$$

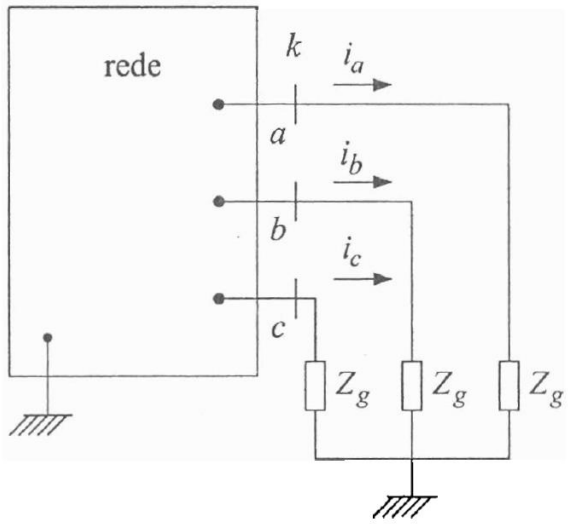
Onde:

i_b = tensão de curto-circuito fase-fase-terra
e_1 = tensão da rede
z_0 = impedância de sequência zero
z_1 = impedância de sequência positiva
z_2 = impedância de sequência negativa

2.1.4 Curto-circuito Fase-Fase-Fase

O curto-circuito fase-fase-fase, também chamado de trifásico, está representado na Figura 4 (BERGEN; VITTAL, 2000).

Figura 4 - Curto-circuito Fase-Fase-Fase



Fonte: Zanetta, 2005, p. 181.

Os curtos-circuitos fase-fase-fase, ou trifásico, são relativamente raros e ocorrem quando as três fases entram em contato. Podem ocorrer quando algum objeto cai sobre a linha, quando algum equipamento, como um transformador, falha ou quando as três fases vão ao solo. Representam 5% das faltas em linhas de transmissão (FAULKENBERRY; COFFER, 1996).

De acordo com Zanetta (2005), a Equação 4 descreve o curto-circuito trifásico.

$$i_{3\phi} = \frac{e_1}{z_1} \tag{4}$$

Onde:

$i_{3\phi}$ = tensão de curto-circuito trifásico
e_1 = tensão da rede
z_1 = impedância de sequência positiva

2.2 Impedância de curto-circuito

Durante uma falta, no ponto em que o condutor entra em contato com o solo ou com outro condutor, normalmente ocorre um arco elétrico. Esse arco é resistivo, porém essa resistência varia amplamente, por isso, a prática é considerá-la zero, de modo a encontrar a máxima corrente de curto-circuito. Para calcular a corrente de curto-circuito mínimo, utiliza-se o valor de 20 Ω. A impedância da falta consiste na resistência do arco elétrico somada a resistência do aterramento, caso a falta esteja relacionada com o solo (FAULKENBERRY; COFFER, 1996).

Existe, porém, a chance da impedância da falta ser muito alta, de modo que não possa ser negligenciada. De acordo com Moreto (2005), esse caso é chamado de Faltas de Alta Impedância (FAI). Devido à alta impedância a falta não gera corrente o suficiente para acionar os relés de sobrecorrente. Essas faltas são muito difíceis de serem detectadas, por causa da baixa magnitude da corrente, que é facilmente confundida com uma simples variação de carga (MORETO, 2005).

As FAI são normalmente causadas pela queda de condutores energizados em superfícies de alta impedância, como solo seco, concreto, cascalho etc. Outra causa comum de FAI é o contato com objetos que estão conectados ao solo, como galhos de árvores. Nesse caso, devido à grande diferença de tensão entre o condutor e o objeto pode ocorre arco elétrico (BAQUI *et al.*, 2011).

Esse tipo de falha não traz riscos para os equipamentos elétricos, contudo são extremamente perigosas para a segurança humana. Logo, a sua detecção se torna uma tarefa crucial para os sistemas de proteção (GHADERI; GINN; MOHAMMADPOUR, 2017).

2.3 Conclusão

Nesse capítulo foi apresentada uma breve revisão da literatura sobre faltas em linhas de transmissão, em que foram detalhados os tipos de curtos-circuitos, como eles ocorrem e sua modelagem matemática através de componentes simétricas. Por fim, foi abordada a impedância de curto-circuito.

No próximo capítulo, é apresentado o referencial teórico sobre redes neurais artificiais.

3 REDES NEURAIS ARTIFICIAIS

Esse capítulo apresenta os fundamentos teóricos sobre redes neurais artificiais.

3.1 Introdução

As Redes Neurais Artificiais (RNA) são redes computacionais que buscam, de forma simplificada, simular a rede de neurônios de um sistema nervoso biológico. Essa simulação difere das estruturas convencionais de computação pois utiliza conceitos fisiológicos de neurônios biológicos e suas formas de organização para a estruturação do algoritmo (GRAUPE, 1997).

Em contraponto com sistemas convencionais que são matematicamente complexos, as RNA proporcionam a resolução de problemas utilizando somente operações matemáticas simples, como adições, multiplicação e lógica simples. Os problemas podem ser complexos, matematicamente mal definidos, não lineares ou estocásticos (GRAUPE, 1997).

Dentre as qualidades das RNA podem se destacar a capacidade de aprendizagem e generalização. A RNA é capaz de aprender sobre o ambiente em que está inserida através da análise de diversos exemplos; esse processo é chamado de treinamento. Além disso, é capaz de extrapolar esse conhecimento e gerar resultados adequados para situações diferentes daquelas de que foi treinada (MORETO, 2005).

Outro aspecto positivo da RNA é o fato de ser, por natureza, um sistema de processamento distribuído. Isso a torna mais robusta a falhas, já que um neurônio defeituoso não compromete o funcionamento geral do sistema. Um exemplo disso é o sistema nervoso de

um adulto em que morrem milhares de neurônios por ano e as funcionalidades não são comprometidas (GRAUPE, 1997).

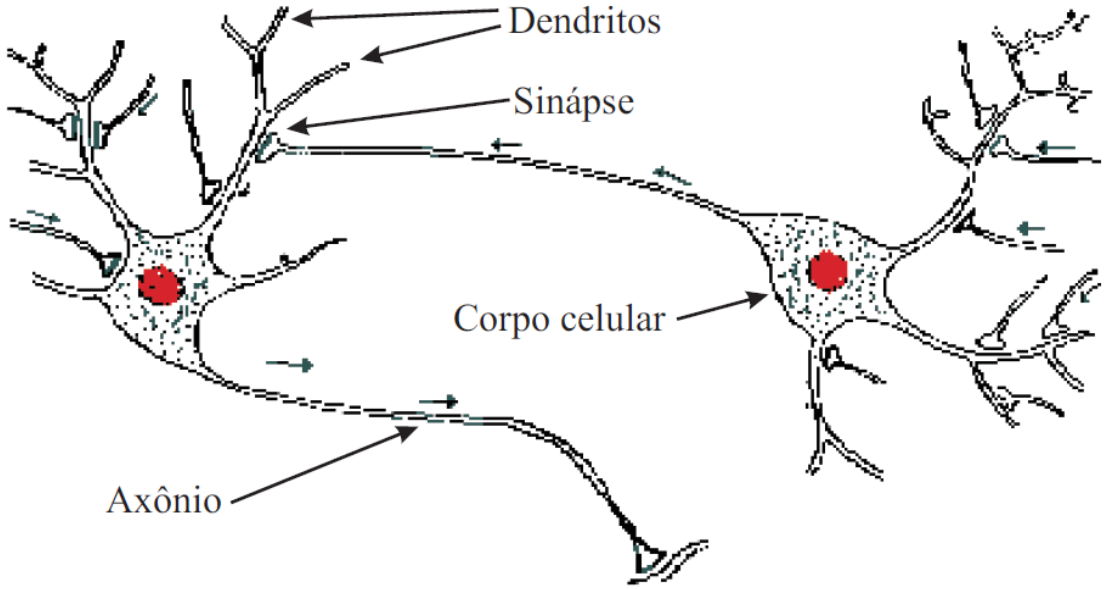
Conforme Kalogirou (2000), as RNA não são indicadas para resolução de tarefas que requerem uma grande precisão, como aplicações de lógica e aritmética. Porém, em diversas outras áreas foram aplicadas com grande sucesso. Ainda, de acordo com Kalogirou (2000), alguns exemplos são:

- Classificação: reconhecimento de padrões, processamento de som e fala, analises de sinais médicos, como eletromiografia;
- Previsão: tendências de mercado, previsão meteorológica, previsão de carga elétrica;
- Sistemas de controle: controles adaptativos e robótica;
- Otimização e apoio a decisão: sistemas de engenharia e de administração.

3.2 O Neurônio Biológico

O neurônio é a base para o funcionamento de redes neurais, sejam elas biológicas ou artificiais. Para que se possa simular um neurônio biológico, é imprescindível conhecê-lo. A Figura 5 ilustra o neurônio biológico, e em destaque, estão as suas partes constituintes, posteriormente descritas, conforme Oliveira (2005), Moreto (2005) e Graupe (1997).

Figura 5 - Neurônio Biológico



Fonte: Moreto, 2005, p. 63.

- Dentritos: região onde o neurônio recebe os sinais de entrada;
- Corpo celular: também chamado de *somma*, é responsável pelo processamento dos sinais de entrada. Quando um certo limiar de entrada é atingido, é disparado um impulso elétrico para a saída do neurônio;
- Axiônio: longo e fino tubo que se fraciona em diversos terminais que quase tocam os dentritos de outros neuôrios. Responsável por transmitir o impulso elétrico disparado pelo *somma*;
- Sinapses: região em que o axiônio de um neurônio (pré-sináptico) entra em contato com o dentrito de um outro neurônio (pós-sináptico). É nessa região em que ocorre a troca de sinis entre neurônios.

O funcionamento do neurônio é baseado em ciclos e constantemente recebe os sinais elétricos dos outros neurônios ligados a ele. A magnitude do sinal recebido depende da eficiência da transmissão da sinapse, seja esse sinal excitatório ou inibitório. Essa eficiência pode ser interpretada como a força da ligação entre os dois neurônios. Quando o neurônio recebe suficentes estímulos elétricos, dentro de um período de tempo, ele dispara um pulso elétrico que viaja pelo axiônio e chega nas sinapses de saída do neurônio excitado. Após o pulso, o neurônio passa por um tempo de recuperação de cargas, em que fica inativo. Quando

recuperado, o neurônio está pronto para começar um outro ciclo de trabalho (MEHROTRA; MOHAN; RANKA, 1996).

O conhecimento apresentado até aqui sobre os neurônios biológicos é somente uma breve introdução de uma grande área de pesquisa, contudo, já é o suficiente para embasar os estudos sobre as redes neurais aritificiais. A seguir, é apresentado o modelo de neurônio artificial, que foi concebido com base no neurônio biológico.

3.3 O Neurônio Artificial

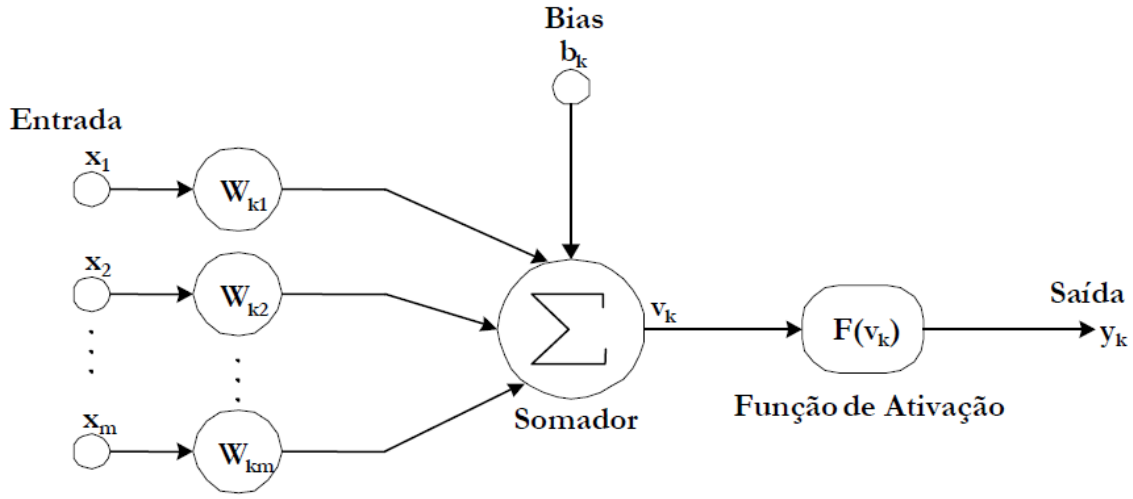
O modelo do neurônio artificial está representado na Figura 6. Ele é basicamente divido em três partes, que são descritas a seguir, conforme Oliveira (2005) e Moreto (2005):

- Sinapses: são as entradas de sinais do neurônio. Cada sinapse possui o seu respectivo peso, que multiplica o sinal de entrada. A função do peso é determinar a importância desse sinal para o neurônio. O valor do peso pode ser positivo, indicando uma sinapse excitatória, ou negativo, indicando uma sinapse inibitória;

- Somador: nessa etapa é realizada a soma ponderada das entradas conforme seus respectivos pesos. Possui, não obrigatoriamente, um polarizador (*bias*, ou *offset*), que pode aumentar ou diminuir a saída do somador para se adequar a função de ativação;

- Função de ativação: função matemática que restringe a saída do somador para uma faixa de valores normalizada, geralmente [-1, 1] ou [0, 1]. Os tipos de funções de ativação mais utilizados estão detalhados na próxima seção desse capítulo.

Figura 6 - Modelo de um neurônio artificial



Fonte: Oliveira, 2005, p. 11.

A Equação 5 apresenta a modelagem matemática do somador, e a Equação 6, a saída do neurônio artificial, segundo Moreto (2005).

$$v_k = \sum_{j=1}^{m} w_{kj} \cdot x_j + b_k \tag{5}$$

$$y_k = \varphi(v_k) \tag{6}$$

Onde:

j = indice da soma

m = número de entradas do neurônio

w_{kj} = pesos das sinapses

x_j = sinal de entrada

b_k = polarização, ou *bias*

v_k = saída do somador, também chamado de campo local induzido

φ = é a função de ativação

y_k = saída do neurônio

É importante ressaltar que o modelo de neurônio artificial atualmente utilizado é primitivo comparada a complexidade existente nos neurônios biológicos (OLIVERIA, 2005).

3.4 Funções de ativação

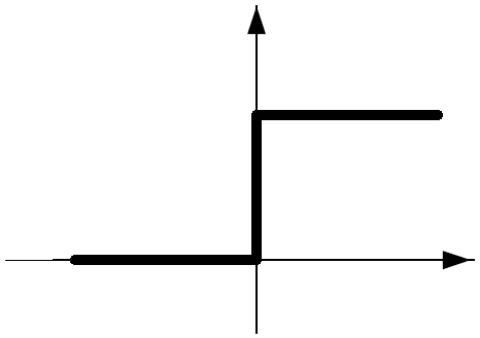
Conforme definido anteriormente, a função de ativação é uma função matemática que restringe a saída do somador para uma faixa de valores normalizada. Cinco exemplos são detalhados nessa seção.

3.4.1 Limiar

Segundo Moreto (2005), a função limiar existe desde o desenvolvimento do neurônio de McCulloch & Pitts, um dos primeiros neurônios artificiais propostos. Essa função é binária, pois possui somente duas saídas: 0 ou 1. Ilustrada na Figura 7, e matematicamente descrita na Equação 7.

$$F(v_k) = \begin{cases} 0, v_k < 0 \\ 1, v_k \geq 0 \end{cases} \quad (7)$$

Figura 7 - Função Limiar



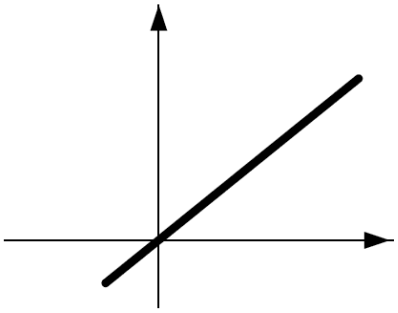
Fonte: Oliveira, 2005, p. 13.

3.4.2 Linear

A função linear está apresentada na Figura 8. Cabe ressaltar que essa função não limita a saída do neurônio. A função matemática equivalente está apresentada na Equação 8.

$$F(v_k) = v_k \quad (8)$$

Figura 8 - Função Linear



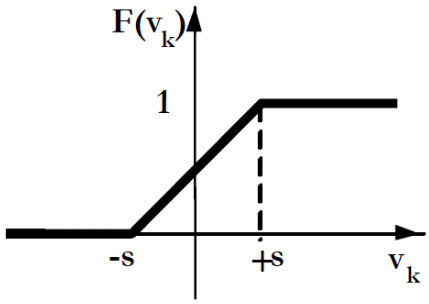
Fonte: Oliveira, 2005, p. 13.

3.4.3 Rampa

A função rampa se difere da linear pois possui limites, inferior e superior, em sua saída. Dentro da faixa [-s, +s], a saída é igual a entrada. Está apresentada na Figura 9, e definida na Equação 9.

$$F(v_k) = \begin{cases} 0, v_k \leq -s \\ \sum_{j=1}^{m} w_{kj} \cdot x_j + b_k \, , -s < v_k < +s \\ 1, v_k \geq +s \end{cases} \quad (9)$$

Figura 9 - Função Rampa



Fonte: Oliveira, 2005, p. 14.

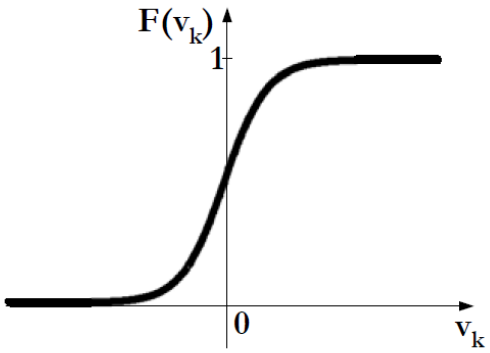
3.4.4 Sigmóide

Segundo Oliveira (2005, p. 14), a função sigmóide é: “uma função semilinear, limitada e monótona. É definida como uma função estritamente crescente, que exibe um balanceamento

adequado entre comportamento linear e não linear". Um exemplo de função sigmóide é a função logística, representada na Figura 10 e descrita na Equação 10.

$$F(v_k) = \frac{1}{1 + e^{(-v_k/T)}} \tag{10}$$

Figura 10 - Função Sigmóide



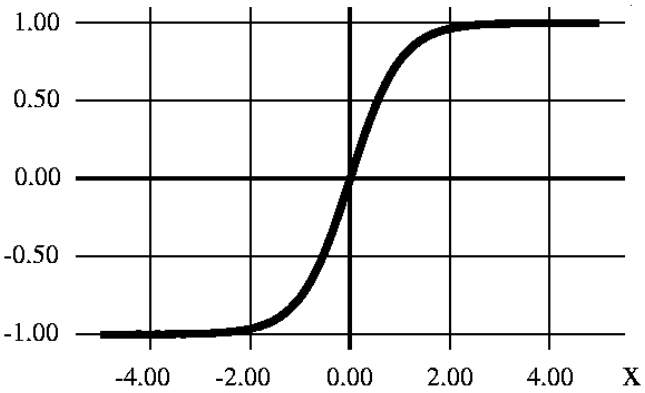
Fonte: Oliveira, 2005, p. 14.

3.4.5 Tangente Hiperbólica

Outro exemplo, conforme Oleskovicz (2001), é a função tangente hiperbólica, ilustrada na Figura 11. A principal diferença entre a função tangente hiperbólica e a sigmóide é que a primeira tem a saída limitada em [-1, 1] e a segunda em [0, 1]. Está descrita na Equação 11.

$$F(v_k) = \frac{e^t - e^{-t}}{e^t + e^{-t}} \tag{11}$$

Figura 11 - Função tangente hiperbólica



Fonte: https://nic.schraudolph.org/teach/NNcourse/figs/tanh.gif <Acessado em 28/10/2017>.

3.5 Arquiteturas de rede

Um único neurônio é insuficiente para resolver algum problema prático, por isso, redes com um grande número de neurônios são frequentemente utilizadas. A forma com que esses neurônios são conectados determina como será constituída a programação da rede, logo, é de suma importância o programador conhecer as arquiteturas disponíveis (MEHROTRA; MOHAN; RANKA, 1996).

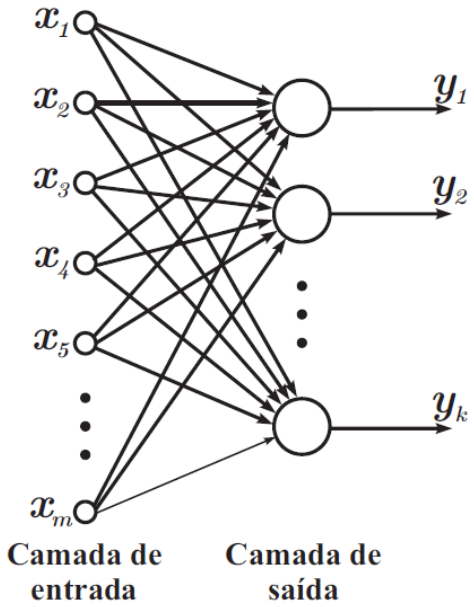
A arquitetura de uma rede neural é definida pela quantidade de camadas de neurônios e pelo tipo de ligação entre eles. De forma geral, existem dois tipos de ligação entre os neurônios: direta (*feed-forward*) e recorrente (*recurrent*). No tipo de ligação direta, o sinal de entrada percorre a rede de maneira unidirecional, sem nenhum tipo de ciclo, já nas redes recorrentes isso não é regra, e as ligações podem ser feitas de maneira arbitrárias, levando a existência de realiamentação de sinal entre os neurônios (RUSSELL; NORVIG, 1995).

As redes neurais do tipo de conexão *feed-forward* são normalmente divididas em camadas de neurônios. Nesse tipo de organização, cada neurônio envia seus sinais somente aos neurônios da próxima camada. Não existem ligações entre neurônios da mesma camada e também ligações que pulem camadas (RUSSELL; NORVIG, 1995).

3.5.1 Rede direta de camada única

As redes de camada única são o modelo mais simples de rede neural. Elas são chamadas de camada única pois possuem somente uma camada de neurônio, como pode ser visto na Figura 12. Essa camada de neurônios é responsável por todo o processamento da rede e também é a saída do sistema (MORETO, 2005). Uma característica importante desse tipo de arquitetura é que ela capaz de classificar somente padrões com classes linearmente separáveis, isso quer dizer que ela não é capaz de solucionar problemas não-lineares (OLIVEIRA, 2005).

Figura 12 - RNA de camada única

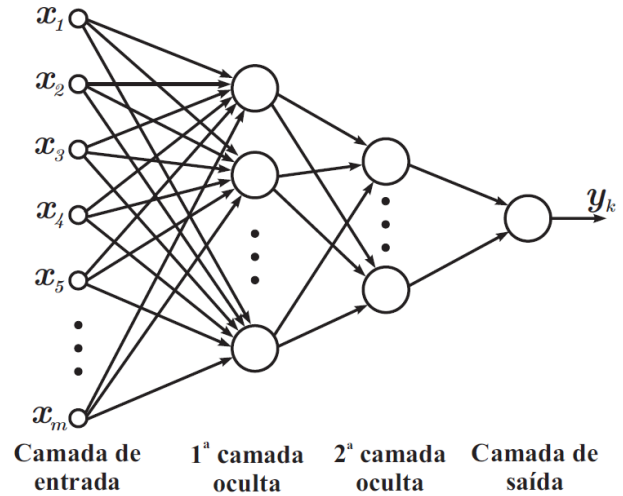


Fonte: MORETO, 2005, p. 66.

3.5.2 Rede direta de múltiplas camadas

As redes de múltiplas camadas se diferenciam das redes de camada simples justamente pela quantidade de camadas de neurônios. Elas possuem uma camada de saída, assim como as redes de camada única, porém possuem também pelo menos uma camada intermediária, também chamada de oculta, entre a entrada e a saída. A Figura 13 ilustra uma RNA de múltiplas camadas, com duas camadas ocultas (MORETO, 2005).

Figura 13 - RNA de múltiplas camadas



Fonte: Moreto, 2005, p. 67.

Conforme Oliveira (2005), as redes de múltiplas camadas com uma camada oculta são capazes de aproximar qualquer função contínua. Já redes com duas camadas ocultas conseguem aproximar qualquer função matemática.

3.6 Paradigmas de aprendizagem

A capacidade de aprender sobre o seu ambiente, e, por consequência, melhorar o seu desempenho, é a característica mais importante das redes neurais artificiais. O aprendizado ocorre através de processos iterativos em que os pesos das sinapses são ajustados de modo que a rede neural consiga chegar a uma solução generalizada para o problema (MORETO, 2005).

Existem diversas formas de conduzir o processo de aprendizagem. A diferença entre os algoritmos de aprendizagem reside na forma com que os pesos sinápticos são alterados. Aplicar um desses algoritmos a uma rede neural significa, de modo geral, mostrar para a rede padrões de informação em um processo iterativo, durante o qual, os pesos das sinapses são ajustados para se obter o menor erro (OLIVEIRA, 2005).

De forma geral, existem dois paradigmas de aprendizagem: a aprendizagem supervisionada e a aprendizagem não-supervisionada, que serão detalhadas a seguir.

3.6.1 Supervisionada

A principal característica desse paradigma de aprendizagem é a existência de um conhecimento prévio sobre as classes que pertencem cada um dos casos analisados durante a fase de treinamento. A diferença entre o valor desejado e o valor calculado pela rede gera um sinal de erro, que é utilizado para ajustar os pesos sinápticos da rede, de modo que a resposta da rede seja a mais próxima possível da esperada (OLESKOVICZ, 2001).

O paradigma de aprendizagem supervisionada, também chamado de aprendizagem com professor, tem o intuito de transferir o conhecimento prévio sobre o ambiente para os pesos sinápticos da rede, de forma que, ao final do processo de treinamento, a rede seja capaz de analisar novos casos, a partir do conhecimento obtido (MORETO, 2005).

3.6.2 Não-Supervisionada

O paradigma de aprendizagem não-supervisionada é caracterizado por não existir um professor para orientar o aprendizado da rede. Esse paradigma é aplicado quando não há um conhecimento prévio sobre o conjunto de dados a ser analisado. A rede é responsável por agrupar os dados de entrada conforme algum critério de similaridade que ela define (OLESKOVICZ, 2001).

O paradigma de aprendizagem não-supervisionada é muito utilizado para a separação de um grande conjunto de dados em conjuntos menores, também chamados de *clusters*, em que cada caso seja parecido com os casos do seu subgrupo e diferente dos casos de outros subgrupos. A aprendizagem não-supervisionada também é utilizada para extrair características (*feature extraction)* de um conjunto de dados. Nesse caso, o objetivo da rede neural é encontrar quais são os aspectos mais importantes, ou seja, com maior variação (MEHROTRA; MOHAN; RANKA, 1996).

3.6.3 O algoritmo de aprendizagem *Backpropagation*

O algoritmo *backpropagation*, ou retropropagação, é utilizado para realizar o ajuste de pesos em redes de múltiplas camadas. É um algoritmo muito popular e vem sendo utilizado para resolver diversos problemas de aprendizagem. Conforme Graupe (1997), o algoritmo foi proposto por Rumenlhart, Hinton e Wiliams, em 1986, porém, já existiam outros trabalhos independentes que propunham algoritmos muito parecidos. O algoritmo *backpropagation* conseguiu resolver o problema de treinamento de redes de múltiplas camadas, uma grande lacuna na época, e alavancou o desenvolvimento de redes neurais artificiais.

O algoritmo consiste em retropropagar o erro (diferença entre a saída desejada e a saída calculada pela rede) da saída da rede neural até a sua entrada, ajustando os pesos sinápticos de forma a minimizar o erro. A complexidade do algoritmo está em determinar quanto se deve alterar cada peso sináptico para que o erro diminua. Para fazer isso, deve ser calculado o quanto cada peso contribuiu para o erro, e ajustá-lo de acordo com essa contribuição e uma taxa de aprendizagem previamente determinada (HAYKIN, 2001).

Conforme Haykin (2001), o algoritmo *backpropagation* pode ser dividido em 5 etapas: inicialização, apresentação dos exemplos de treinamento, propagação, retropropagação e iteração.

A primeira etapa consiste em inicializar os pesos sinápticos da rede. Os pesos devem ser inicializados com valores aleatórios, retirados de uma distribuição uniforme com média zero. É importante que a faixa de variação dos pesos comtemple a faixa de transição entre as partes linear e saturada da função de ativação utilizada (HAYKIN, 2001).

A segunda etapa consiste em apresentar um conjunto de exemplos de treinamento para a rede. Para cada exemplo apresentado, devem ser realizadas as etapas 3 e 4, respectivamente. Já a terceira etapa consiste na propagação dos sinais de entrada (um exemplo do conjunto de treinamento) através da rede neural. Deve ser calculado a saída de cada neurônio, camada por camada, até que se chegue nas camadas de saída (HAYKIN, 2001). Primeiramente deve ser calculado o campo local induzido, descrito na Equação 12.

$$v_j^{(l)}(n) = \sum_{i=0}^{m} w_{ij}^{(l)}(n) y_i^{(l-1)}(n) \tag{12}$$

Onde:
n = número da iteração
l = número da camada
j = número do neurônio na camada
m = tamanho da camada l
$w_{ij}^{(l-1)}$ = peso da sinapse
$y_i^{(l-1)}$ = saída do neurônio i da camada anterior

Após, a saída do neurônio, descrita na Equação 13.

$$y_j^{(l)} = \varphi_j\left(v_j^{(l)}(n)\right) \tag{13}$$

Onde:
$y_j^{(l)}$ = saída do neurônio
φ_j = função de ativação
$v_j^{(l)}$ = campo local induzido do neurônio

Caso o neurônio faça parte da última camada, a sua saída equivale a saída da rede. Por fim, o erro deve ser calculado conforme a Equação 14.

$$e_j(n) = d_j(n) - o_j(n) \tag{14}$$

Onde:
e_j = erro
d_j = saída desejada
o_j = saída calculada

A quarta etapa consiste em propagar o erro de volta, atualizando os pesos sinápticos no caminho. Deve ser calculada a influência de cada peso no erro. Para isso, é calculado o gradiente local da rede. Para neurônios da camada de saída, é utilizada a Equação 15.

$$\delta_j(n) = e_j(n)\varphi_j'\left(v_j(n)\right) \tag{15}$$

Onde:
φ' = derivada da função de ativação
δ_j = gradiente local

Para neurônios de camadas ocultas, é utilizada a Equação 16:

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\sum_k \delta_k(n) w_{kj}(n) \tag{16}$$

Onde:
k = neurônios da camada a direita do neurônio j

Determinado o gradiente local de cada neurônio, seus pesos são ajustados conforme a Equação 17.

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta\, \delta_j^{(l)}(n)\, y_i^{(l-1)}(n) \tag{17}$$

Onde:
η = taxa de aprendizagem

Por fim, a quinta etapa consiste em iterar a terceira e a quarta etapa através de todos os exemplos do conjunto. Após isso, é calculado o erro médio quadrático e, caso ele estiver estabilizado, ou menor do que um determinado valor, então o treinamento está pronto. Caso o

resultado ainda não for o esperado, volta-se a segunda etapa, onde são apresentados os mesmos exemplos, porém em uma ordem diferente, que pode ser aleatória.

3.7 Treinamento de Redes Neurais Artificiais

Durante o desenvolvimento do trabalho foi percebida a necessidade de definir noções relacionados ao treinamento de RNA. Nessa seção são apresentadas essas noções e os seus significados segundo a bibliografia.

3.7.1 Validação Cruzada e *k-fold*

Um problema grave que pode ocorrer em qualquer processo de *machine learning*, seja ele através de redes neurais artificiais ou não, é o *overfitting*. Durante o processo de treinamento existe a probabilidade do modelo se adequar excessivamente ao conjunto de dados a ele apresentado, consequentemente, perdendo a sua capacidade de generalização (HAYKIN, 2001).

A validação cruzada é uma técnica utilizada para diminuir os efeitos do *overfitting*. Ela consiste em dividir o conjunto total de dados em subconjuntos de treinamento e validação. O primeiro servirá somente para o treinamento do modelo, e o segundo servirá somente para avaliar o desempenho. Dessa forma, se o modelo estiver excessivamente adequado ao conjunto de treinamento, o seu desempenho ao analisar o conjunto de validação será menor (RUSSEL e NORVIG, 1995).

Além da validação cruzada, outra técnica utilizada para melhor avaliar o desempenho do modelo é a *k-fold*. Essa técnica consiste em dividir o conjunto de dados em *k* conjuntos do mesmo tamanho, e, de forma iterativa, usar um dos *k* conjuntos para validação e ou outros como treinamento. Dessa forma, o modelo será avaliado *k* vezes de maneira diferente, diminuindo a probabilidade do *overfitting* (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.7.2 *Batch Size*

O *batch size*, também chamado de *minibatch*, é o número de exemplos que são processados antes de realizar a atualização dos pesos da rede neural artificial. Esse número

normalmente é maior que 1 e menor do que o tamanho do conjunto de exemplos (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.7.3 Épocas

Conforme Haykin (2001), em aplicações práticas do algoritmo *backpropagation*, o aprendizado da rede é fruto de muitas apresentações do conjunto de treinamento para a rede neural artificial. Russel e Norvig (1995) complementam, definindo que uma apresentação completa de todos os exemplos do conjunto de treinamento é denominada de época.

3.7.4 Função Perda

A função perda (*loss function*), também chamada de função erro (*error function*) ou de função de custo (*cost function*), é a função utilizada para calcular o erro da saída da rede neural artificial durante o processo de treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.7.5 Otimizadores

Conforme Braga, Carvalho e Ludermir (2011), a principal dificuldade encontrada no processo de treinamento de uma rede neural artificial através do algoritmo *back-propagation* é a sua sensibilidade às características da superfície de erro do conjunto de dados. A convergência do algoritmo fica comprometida em regiões de mínimos locais e baixos gradientes. De modo a resolver esse problema, algumas ações podem ser tomadas como: utilizar taxas de aprendizagem decrescente, utilizar um termo *momentum* e adicionar ruído ao conjunto de dados.

Para contornar problemas dessa origem são utilizados algoritmos otimizadores que procuram pelos parâmetros de aprendizado que minimizam a saída da função perda. A principal finalidade dos otimizadores é aumentar a eficiência do treinamento das redes neurais artificiais (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.8 Conclusão

Nesse capítulo foi apresentada uma revisão da literatura sobre as redes neurais artificiais. Iniciou-se com uma introdução sobre as redes neurais, evidenciando suas aplicações e foram apresentados o neurônio biológico e o neurônio artificial. Após isso, foram apresentadas as arquiteturas de rede e os paradigmas de aprendizagem, elucidando o algoritmo *backpropagation*. Por fim, foram abordados conceitos importantes sobre o treinamento de rede neurais.

No próximo capítulo, é apresentado o desenvolvimento do projeto.

4 DESENVOLVIMENTO

O objetivo geral do trabalho é desenvolver um sistema de detecção e classificação de faltas em linhas de transmissão utilizando redes neurais artificiais. Para atingir esse objetivo, o desenvolvimento foi dividido nas seguintes etapas:

a) Modelagem da linha de transmissão;

b) Criação de uma base de dados de simulações;

c) Implementação da rede neural artificial;

d) Ajuste fino dos parâmetros da rede neural artificial.

Nesta seção está descrito o processo de desenvolvimento de cada uma dessas etapas, começando com uma compilação das tecnologias utilizadas.

4.1 Tecnologias utilizadas

Para a criação do banco de dados de faltas, as seguintes ferramentas foram utilizadas:

a) ATP: Acrônimo de *Alternate Transients Program*, o ATP possui um conjunto de ferramentas para simulação e análise de transitórios eletromagnéticos em sistemas elétricos de potência. Foi utilizado para a modelagem do circuito elétrico (FARIAS, 2017);

b) GTPPLOT: Ferramenta para plotagem e conversão de arquivos de simulação do ATP. Foi utilizado para converter os arquivos do formato PL4 para um formato legível (EMPT-ATP, 2018);

c) PlotXY: Ferramenta de plotagem de arquivos de simulação do ATP em ambiente gráfico Windows. Foi utilizado para verificar os resultados das simulações geradas (EMPT-ATP, 2018);

d) Java: É uma linguagem de programação e uma plataforma de *software*. A linguagem é caracterizada por ser simples, orientada a objetos, multiplataforma, robusta e segura. A plataforma é baseada na *Java Virtual Machine* (JVM), uma máquina virtual capaz de executar código Java em diversos sistemas operacionais. Foi utilizado para desenvolver o algoritmo de geração das simulações (ORACLE, 2018);

e) Cassandra: É um Sistema de Gerenciamento de Banco de Dados (SGBD) de código aberto, mantido pela *Apache Software Foundation*. O Cassandra é um banco de dados NoSQL, desenvolvido na linguagem de programação Java, descentralizado, altamente escalável e com capacidade de gerenciar um grande volume de dados. Foi utilizado para armazenar os resultados das simulações (CASSANDRA, 2018);

f) Docker: Plataforma de containers. Automatiza tarefas de configuração e instalação de sistemas operacionais e aplicações. Foi utilizado para executar o SGBD Cassandra (DOCKER, 2018);

g) MySQL Workbench: Conjunto unificado de ferramentas visuais para arquitetos de banco de dados, desenvolvedores e administradores de sistemas. Possui ferramentas de modelagem de dados, desenvolvimento de *Structured Query Language* (SQL) e de gerenciamento de bancos de dados. Foram utilizadas as ferramentas de modelagem para o desenvolvimento do banco de dados de faltas (MYSQL, 2018).

No desenvolvimento da rede neural artificial, as seguintes ferramentas foram utilizadas:

a) Python: É uma linguagem de programação interpretada e orientada a objetos. Possui estruturas de dados altamente dinâmicas e uma sintaxe muito clara. Pode ser estendida em C ou C++. É multiplataforma, sendo possível executa-la em Unix, Mac e Windows (PYTHON, 2018);

b) Anaconda: É uma plataforma de desenvolvimento Python. Possui código fonte aberto, é multiplataforma e focada em *data science*. Oferece um conjunto completo de ferramentas de análise de dados e possui uma fácil instalação (ANACONDA, 2018);

c) Keras: É uma *Application Programing Interface* (API) de alto-nível para desenvolvimento de redes neurais. Foi desenvolvida em Python com o foco de proporcional ao programador uma rápida experimentação (CHOLLET *et al.*, 2015);

d) Scikit-Learn: É uma biblioteca para a linguagem Python que fornece um grande conjunto de ferramentas para mineração e análise de dados. Possui código fonte aberto (PEDREGOSA *et. al.*, 2011);

e) Tensorflow: É uma biblioteca de código aberto para computação numérica de alta performance. Pode ser utilizada em *desktops* ou em *clusters* e roda tanto em *Central Processing Unit* (CPU) quanto em *Graphics Processing Unit* (GPU). Foi utilizada como *backend* para a biblioteca Keras (TENSORFLOW, 2018).

O sistema operacional utilizado para a modelagem do circuito e da criação da base de dados de falta foi o Windows, por motivos de compatibilidade das ferramentas necessárias (ATP, GTPPLOT e PlotXY). A implementação e testes da rede neural artificial foram feitos no sistema operacional Ubuntu, uma distribuição Linux gratuita. A versão de cada uma das ferramentas e tecnologias utilizadas estão descritas na Tabela 1.

Tabela 1 - Versões das tecnologias utilizadas

Ferramenta	**Versão**	**Ferramenta**	**Versão**
Ubuntu	16.04	Docker	18.03.1-ce-win65
Windows	10	MySQL Workbench	6.3.6
ATP	14-Apr-2010	Python	3.5.5
GTPPLOT	29-Apr-2010	Anaconda	1.8.5
PlotXY	Nov-2010	Keras	2.1.6
Java	1.8.0_102	Scikit-Learn	0.19.1
Cassandra	3.11.2	Tensorflow	1.8.0

Fonte: Autor

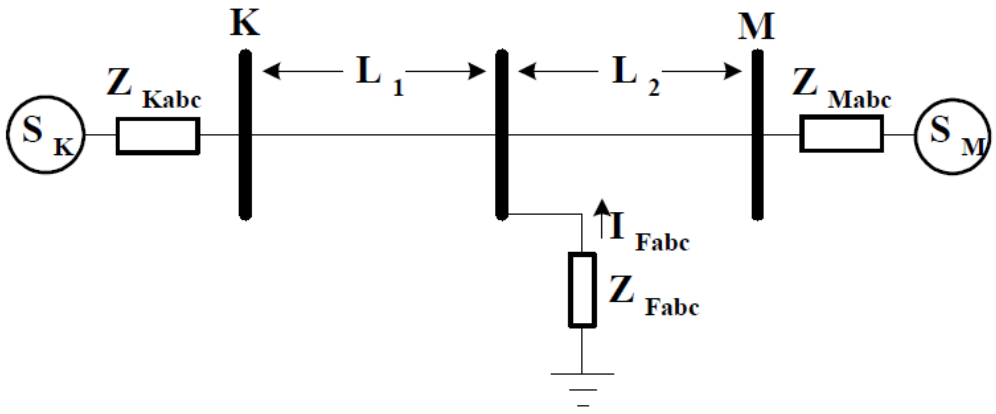
4.2 Modelagem da linha de transmissão

Para realizar o treinamento da rede neural artificial foi necessário criar uma base de dados de simulação de faltas. Para isso, o primeiro passo foi modelar computacionalmente um circuito capaz de simular faltas em linhas de transmissão.

O circuito adotado, apresentado na Figura 14, consiste em uma linha de transmissão com um equivalente em cada um dos seus terminais, conforme utilizado por Oliveira (2005). Entre as barras K e M existe uma barra para simulação da falta, que divide a linha de transmissão em L1 e L2.

A Tabela 2 apresenta os parâmetros de sequência zero e positiva utilizado na modelagem das linhas de transmissão. A Tabela 3 apresenta as impedâncias de sequência zero e positiva das fontes e, por fim, a Tabela 4 apresenta a tensão e ângulo das fontes. A tensão base é 400 kV.

Figura 14 – Modelo de linha de transmissão desenvolvido



Fonte: Oliveira, 2005, p. 43.

Tabela 2 - Parâmetros das linhas de transmissão

Comprimento da Linha	200 milhas
R_0	0.249168 Ω/milha
L_0	0.60241 Ω/milha
C_0	1.56277 mH/milha
R_1	4.8303 mH/milha
L_1	19.469 nF/milha
C_1	12.06678E nF/milha

Fonte: Adaptado de Oliveira (2005) e Brahma e Girgis (2004).

Tabela 3 – Impedância das fontes

Z_{SK0}	17.177 + j45.5285 Ω
Z_{SK1}	2.5904 + j14.7328 Ω
Z_{SM0}	15.31 + j45.9245 Ω
Z_{SM1}	0.7229 + j15.1288 Ω

Fonte: Adaptado de Oliveira (2005) e Brahma e Girgis (2004).

Tabela 4 - Tensões nas barras

V_{SK0}	408248∠20,00º V
V_{SK1}	408248∠00,00º V

Fonte: Adaptado de Oliveira (2005) e Brahma e Girgis (2004).

O *software* escolhido para a modelagem do circuito foi o ATP. O Apêndice A apresenta a descrição do circuito, através de um arquivo de texto no formato de entrada do *software*. As linhas de transmissão foram modeladas utilizando o componente RLC mutuamente acoplado transposto, com os parâmetros apresentados na Tabela 2. As impedâncias das fontes foram modeladas utilizando o componente RL mutuamente acoplado, conforme os dados da Tabela 3.

Na barra da falta foi elaborado um conjunto de *switches* (chaves) capaz de simular todos os tipos de faltas. Cada fase possui um switch que liga ela à terra, através de uma resistência, e dois *switches* que ligam às outras duas fazes, sendo assim possível reproduzir faltas monofásicas, bifásicas e trifásicas.

A frequência base escolhida para a simulação foi de 60 Hz. A frequência de aquisição utilizada foi de 6000 Hz, resultando em aproximadamente 100 pontos por ciclo. O tempo total da simulação escolhido foi de 0,1 segundos, resultando num total de 600 pontos por simulação.

4.3 Criação de uma base de dados de simulação

A partir da modelagem do circuito foi necessário gerar simulações, cada uma com parâmetros randômicos, visando abranger diversos casos de operação. Com a finalidade de agilizar o processo de criação dessa base, foi implementado um algoritmo que automatiza o processo de geração e armazenamento das simulações. Os seguintes parâmetros foram alterados em cada simulação:

- Tipo do curto-circuito: Fase-Fase, Fase-Terra, Fase-Fase-Terra, Fase-Fase-Fase;
- Ângulo de incidência da falta: Ângulo de defasagem da ocorrência da falta;
- Local da falta: Distância entre o início da linha e o ponto da falta;
- Impedância da falta: Impedância entre os condutores envolvidos e/ou o terra.

O algoritmo desenvolvido realiza os seguintes passos, de maneira iterativa:

a) Lê o arquivo de simulação base, no formato ATP;

b) Gera os parâmetros para a simulação atual;

c) Grava um arquivo temporário com os parâmetros gerados;

d) Gera a simulação com o *software* ATP, com os dados do arquivo temporário;

e) Converte o resultado do formato PL4 para CSV;

f) Insere os resultados em um banco de dados.

O modelo do banco de dados, chamado de *keyspace* no SGBD Cassandra, possui duas tabelas: uma para armazenar os dados gerais sobre cada simulação (*fault_case*) e outra para armazenar os pontos da simulação, juntamente com a indicação do tipo de falta (*fault_case_data*). A modelagem das tabelas está apresentada na Figura 15, na página 1645.

É importante ressaltar que as tabelas estão relacionadas pelo campo *fault_case_id*, porém, somente de forma lógica, visto que o SGBD Cassandra não suporta chaves estrangeiras.

Como somente os dados da tabela *fault_case_data* são utilizados para o treinamento da rede neural, a falta do suporte a chaves estrangeiras não interfere no desempenho do modelo.

A modelagem da tabela *fault_case_data* possui o tempo total decorrido da simulação, a corrente e a tensão trifásicas instantâneas medidas na barra K, que são as entradas da rede neural aritificial, e 12 campos booleanos que indicam o estado do sistema.

O primeiro campo booleano, *fault*, indica se o estado atual do sistema é normal ou faltoso. Esse campo foi utilizado somente na implementação inicial da rede neural. Os outros 11, indicam qual é o tipo da falta que está ocorrendo. A tabela foi modelada dessa forma para corresponder à saída de redes neurais de classificação, que possuem um neurônio para cada classe existente no conjunto de dados. Dessa maneira, não serão necessárias outras formas de pré-processamentos dos dados para utilizá-los na rede neural.

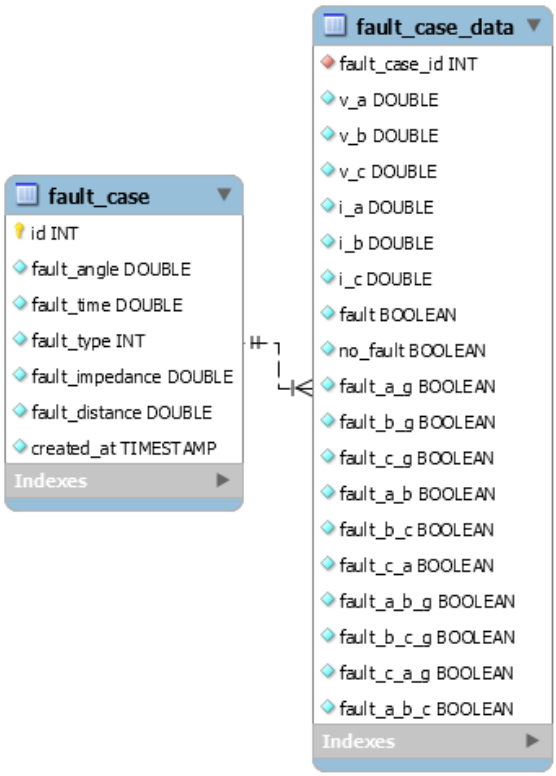
Após finalizar o algoritmo foi gerado uma base de dados com 549 casos de falta, totalizando em 549549 pontos de simulação para treinamentos e testes da rede neural. Os parâmetros da simulação foram gerados aleatoriamente dentro das seguintes faixas:

- Tipo do curto-circuito: AG, BG, CG, AB, BC, CA, ABG, BCG, CAG e ABC;
- Ângulo de incidência da falta: Entre 0º e 360º;
- Local da falta: Entre 1 e 199 milhas;
- Impedância da falta: Entre 1 e 100 ohms.

Além dos parâmetros da simulação, o próprio algoritmo possui um arquivo de configuração, onde são inseridas as informações de conexão com o banco de dados, diretórios de trabalho, caminho do arquivo base, caminho dos executáveis do ATP e do GTPPL e o número de simulações a criar. Dessa forma, o usuário pode alterar os valores sem a necessidade de acessar o código fonte e recompilar o sistema.

Visando facilitar o manuseio dos dados, foi desenvolvido uma rotina em Python utilizando as bibliotecas Pandas e SciKit-Learn. Essa rotina conecta no banco de dados das simulações, converte o conjunto de dados para o formato DataFrame da biblioteca Pandas e exporta para um arquivo no formato *pkl* utilizando o módulo *joblib* da biblioteca SciKit-Learn. Dessa forma o banco de simulações ficou contido em um único arquivo, facilmente importado por outras rotinas Python.

Figura 15 - Modelo do banco de dados de faltas



Fonte: Autor.

4.4 Implementação da Rede Neural Artificial

Em um primeiro momento, foi desenvolvida uma rede neural somente para a detecção de faltas, sem classifica-las. Dessa forma, com menos dados e menos complexidade, o tempo de treinamento foi menor, proporcionando uma maior eficiência na programação. O objetivo era implementar um modelo que conseguisse convergir e aprender o padrão, sem preocupação com a precisão final.

O primeiro passo foi determinar quais são os parâmetros necessários para definir a rede neural artificial. Os seguintes parâmetros foram elencados:

- Função de custo;
- Função de ativação;
- Otimizador;
- Taxa de aprendizagem;
- Número de camadas intermediárias;
- Número de neurônios nas camadas intermediárias.

Além dos parâmetros da rede neural, outros parâmetros foram necessários para o processo de treinamento. São eles:

- Tamanho do conjunto de teste;
- Número de épocas;
- *Batch Size*.

De forma geral, a rotina implementada segue os seguintes passos:

a) Carrega o arquivo contendo a base de dados de faltas;

b) Separa os valores da base em entrada (X) e de saída (Y);

c) Divide os dados em conjunto de treinamento e conjunto de testes;

d) Define os parâmetros da rede neural artificial;

e) Compila a rede neural, com os parâmetros definidos;

f) Executa o treinamento da rede, com o conjunto de dados de treinamento;

g) Estima a precisão da rede, através do conjunto de dados de teste;

h) Converte o modelo treinado e salva em um arquivo no disco rígido.

Após chegar a um modelo que conseguiu detectar as faltas, foram feitas modificações para que ele pudesse classifica-las. A topologia da rede foi alterada, passando de uma saída para 11 saídas, e foi aumentado o número de camadas intermediárias e de neurônios. Da mesma forma que a implementação anterior, o objetivo foi de chegar a um modelo que convergisse.

4.5 Ajuste fino dos parâmetros da rede neural artificial

A partir da rede neural artificial implementada na etapa anterior, foram feitos testes para determinar os parâmetros que levassem o modelo a obter a maior precisão possível. Neste subitem são apresentados os ajustes realizados e, por fim, a rede neural resultante.

Todos os ajustes dessa seção foram realizados com validação cruzada e *k-fold*. O fator *k* escolhido para os testes foi de 20.

A seguir, estão apresentados todos os ajustes realizados.

4.5.1 Função Custo

Para determinar a função custo ideal para a rede neural artificial, foram realizados testes de precisão com 5 funções diferentes. Cada uma das funções foi testada com 50, 80 e 110 épocas. As funções testadas foram a *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE), *Root Mean Squared Error* (RMSE), Logaritmo do cosseno hiperbólico (LOGCOSH) e C*ross-entropy* (CROSS). A Figura 16 apresenta um gráfico com o resultado dos testes realizado, coma precisão média encontrada para cada função custo utilizada.

A função custo que resultou na maior precisão da rede neural artificial foi a *cross-entropy* (CROSS), chegando a 91,669% com 80 épocas de treinamento. A função *cross-entropy* busca diminuir o tempo de aprendizagem levando em consideração o fato de que, quanto maior o erro, muito maior deve ser a correção dos pesos sinápticos, dessa forma o custo do erro cresce de forma exponencial, conforme a Equação 18, apresentada por Nielsen (2015).

$$C = -\frac{1}{n}\sum_{x}[y \ln a + (1-y)\ln(1-a)] \tag{18}$$

Onde:

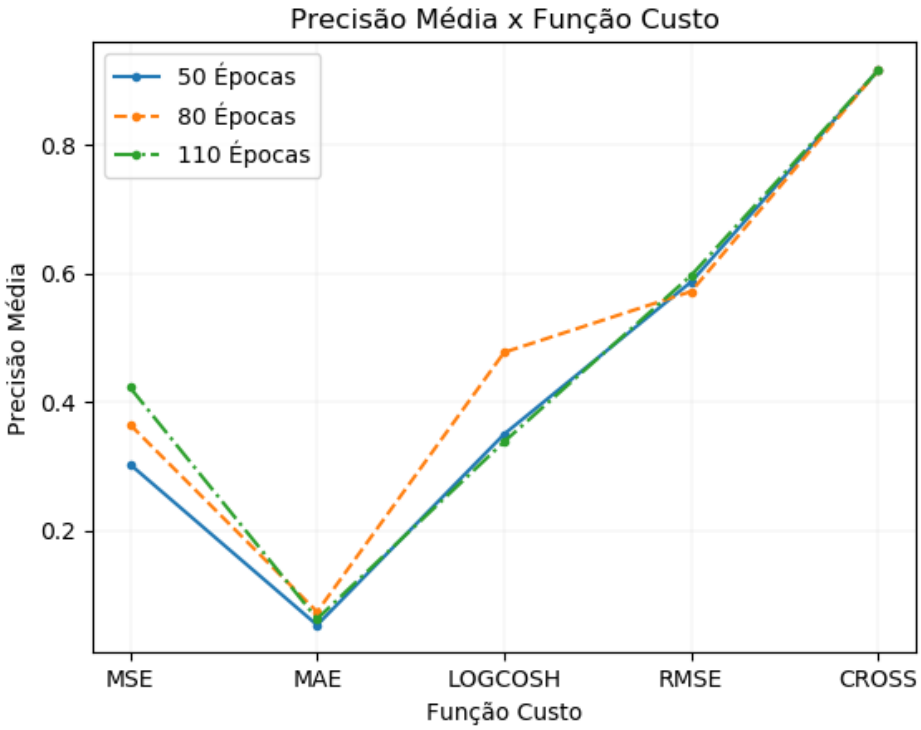
n = número de itens no lote de treinamento

x = item do lote de treinamento

y = saída desejada

a = saída calculada

Figura 16 – Precisão Média x Função Custo

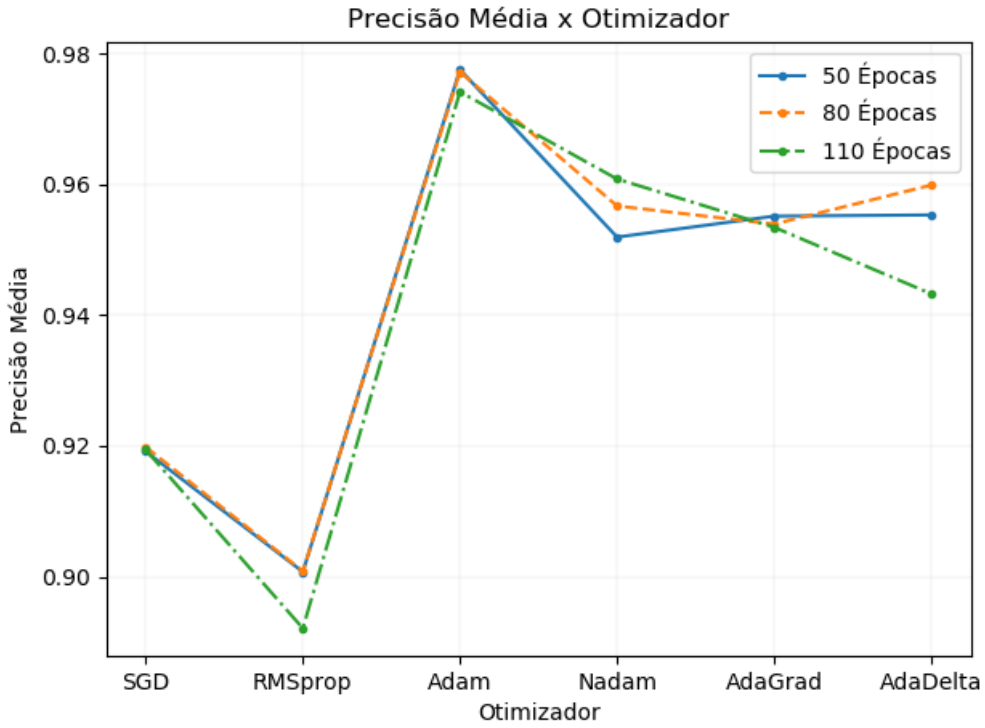


Fonte: Autor.

4.5.2 Otimizador

A determinação do algoritmo otimizador ocorreu de forma semelhante à da função custo. Foram elencados 6 algoritmos e testado a precisão de cada um em 50, 80 e 110 épocas. Os otimizadores testados foram: SGD, RMSprop, Adam, Nadam, AdaGrad e AdaDelta. A Figura 17 apresenta o resultado dos testes, em um gráfico que mostra a precisão média final da rede neural em cada um dos testes. O otimizador que resultou na maior precisão média da rede neural artificial foi o Adam, chegando a 97,75% em 50 épocas de treinamento.

Figura 17 - Precisão Média x Otimizador



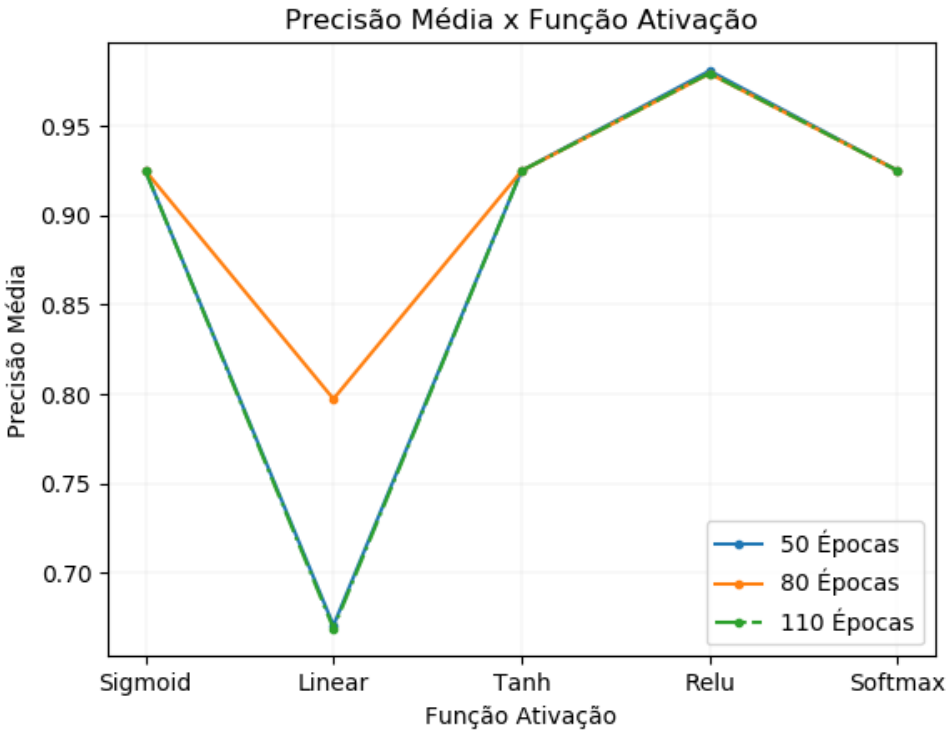
Fonte: Autor

Proposto por Kingma e Ba (2015), o algoritmo de otimização de redes neurais Adam é um otimizador estocástico de primeira ordem de gradiente, baseado em estimativas adaptáveis de momentos de baixa ordem. Surgiu da combinação das vantagens de outros dois otimizadores, o RMSprop e o AdaGrad.

4.5.3 Função de ativação

Para determinar a função de ativação das camadas intermediárias foram elencadas 5 funções e realizados testes de precisão com 50, 80 e 110 épocas. As funções escolhidas para o teste foram a sigmoide, a linear, a tangente hiperbólica, a *Rectified Linear Units* (ReLU) e a *Softmax*. O resultado dos testes está apresentado na Figura 18.

Figura 18 - Precisão Média x Função Ativação



Fonte: Autor

O resultado do teste mostrou que as funções de ativação sigmoide, tangente hiperbólica e *Softmax* acabaram resultando em precisões muito semelhantes, independentemente do número de épocas. A função linear teve um resultado consideravelmente pior comparado as outras, apresentando uma melhora somente em 80 épocas.

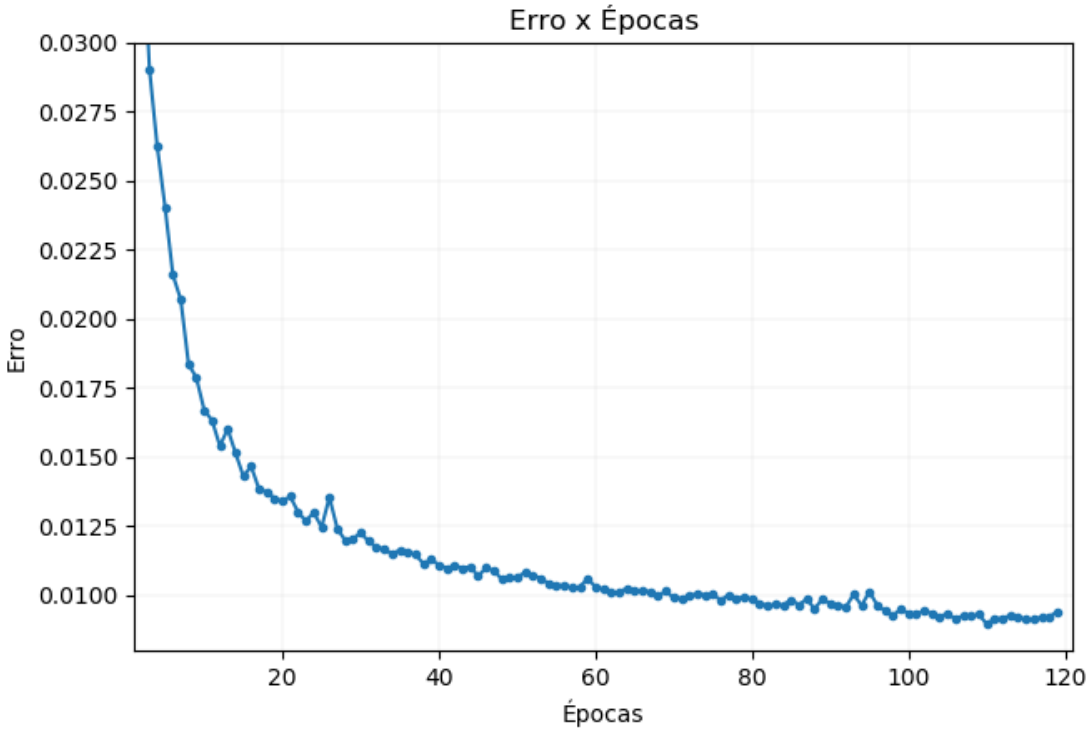
A função de ativação que apresentou o melhor desempenho, independentemente do número de épocas de treinamento, foi a função ReLU. A ReLU é a função de ativação não linear e simples: se o valor de entrada é menor que zero, a sua saída é zero, caso contrário, a sua saída é igual ao valor de entrada (NAIR; HINTON, 2010). A Equação 19 descreve o seu funcionamento.

$$f(x) = \max(x, 0) \tag{19}$$

4.5.4 Número de Épocas

O ajuste do número de épocas foi feito através da análise do gráfico de erro da rede neural artificial versus a época de treinamento, apresentado na Figura 19.

Figura 19 - Erro x Épocas



Fonte: Autor

Foi verificado que a partir da época número 100 o erro não tem mais uma tendência de diminuição. A média do erro entre as épocas 100 e 120 ficou em 0,00924. Dessa forma, o número de épocas de treinamento adotado foi de 110.

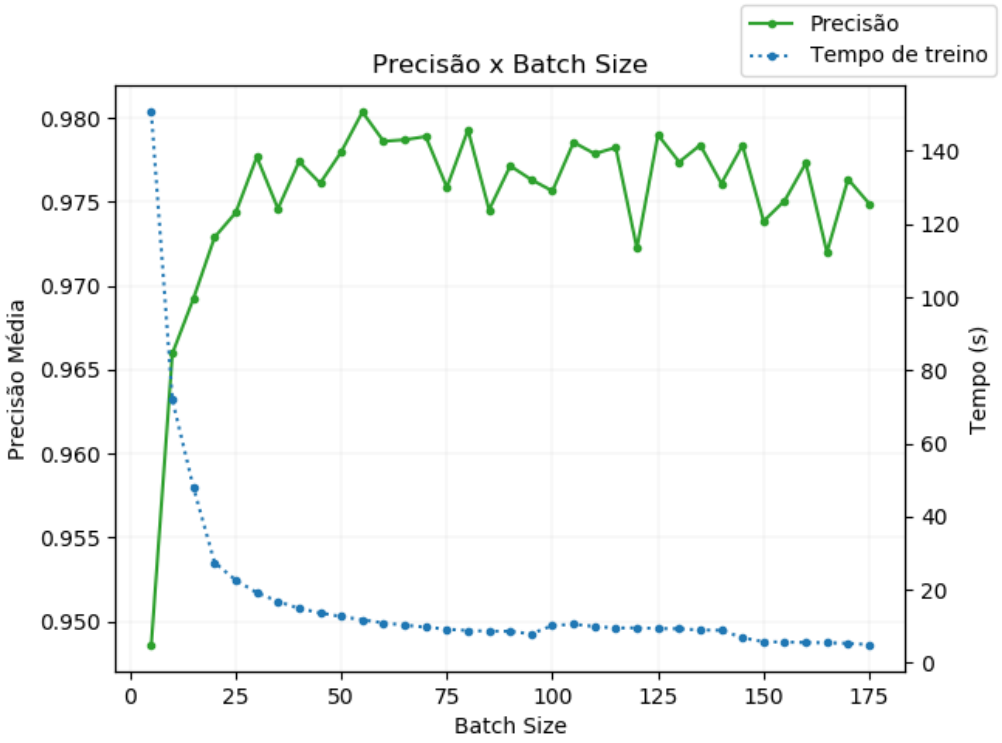
4.5.5 *Batch Size*

A escolha do *batch size* foi feita através de uma varredura, verificando qual o valor que resultaria na maior precisão de rede neural artificial. O *batch size* foi variado de 5 a 180, em intervalor de 5 unidades. O resultado do teste está apresentado no gráfico da Figura 20.

O *batch size* que resultou na maior precisão foi 55, chegando a 98,03%. É importante ressaltar o quanto esse parâmetro influencia no tempo de treinamento da rede. A linha azul tracejada do gráfico da Figura 20 apresenta o tempo médio de treinamento para a rede versus o *batch size*, que varia de 4,9 segundos no melhor caso até 150,6 segundos no pior. É possível notar também que o tempo decresce exponencialmente com o aumento do *batch size*, logo

valores pequenos, menor que 20, não são aconselhados pois elevam consideravelmente o tempo de treinamento.

Figura 20 - Precisão x *Batch Size*



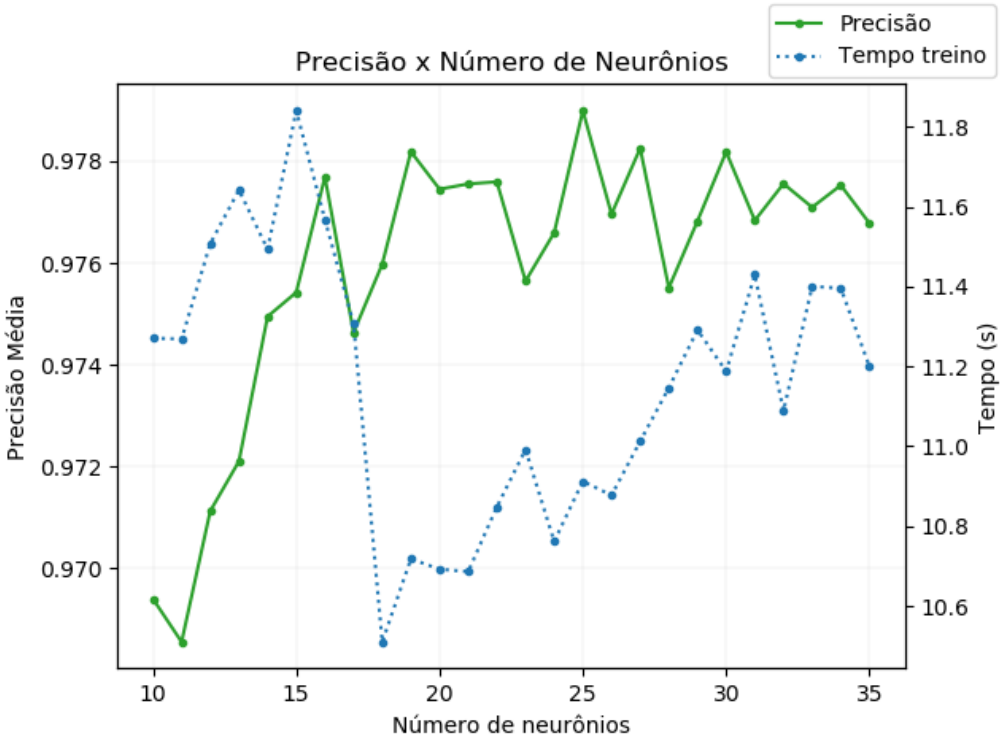
Fonte: Autor

4.5.6 Número de neurônios nas camadas intermediárias

O ajuste do número de neurônios nas camadas intermediárias foi feito através de uma varredura, identificando o parâmetro que resultasse na maior precisão média da rede neural artificial. A varredura começou em 10 neurônios e foi até 35, incrementando unitariamente. O resultado está apresentado na Figura 21.

A precisão chegou ao pico de 97,9% com o número de neurônios em 25, logo, esse foi o valor adotado. No gráfico também está apresentado o tempo médio de treinamento, que variou entre 11,84 e 10,51 segundos. Isso mostra que, nessa aplicação, o número de neurônios nas camadas intermediárias não influenciou de forma significativa o tempo de treinamento da rede neural.

Figura 21 - Precisão x Número de neurônios nas camadas intermediárias



Fonte: Autor

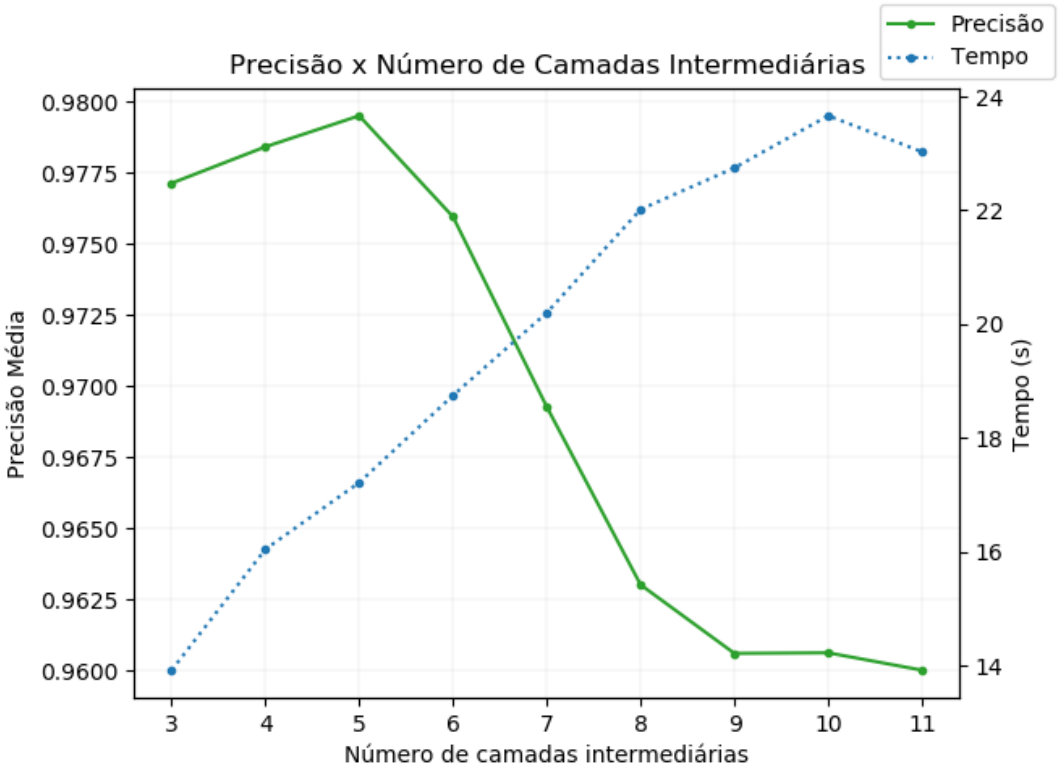
4.5.7 Número de camadas intermediárias

A técnica utilizada para a determinação do número de camadas intermediárias foi a de crescimento da rede. Conforme Haykin (2001), essa técnica consiste em começar com uma rede pequena, com poucas camadas e adicionar camadas a medida que a rede neural artificial não conseguir mais resolver o problema.

Essa técnica foi escolhida visando diminuir problemas de convergência. Conforme Braga, Carvalho e Ludermir (2011) a utilização de mais camadas intermediárias do que o necessário pode levar a problemas de convergência, visto que essas camadas atualizam os pesos de seus neurônios através da estimativa de erro, e quando maior o número de camadas, mais difícil é a convergência.

Dessa forma, iniciou-se com 3 camadas intermediárias e foram adicionadas gradativamente mais camadas e calculando a precisão média do modelo. O resultado do teste está apresentado na Figura 22. O número de camadas intermediárias que resultou na maior precisão foi 5, chegando a 97,95%.

Figura 22 - Precisão x Número de camadas intermediárias



Fonte: Autor

O gráfico apresenta também o tempo médio de treinamento da rede neural artificial. É possível verificar que o tempo aumenta linearmente com o aumento do número de camadas. Ao utilizar 5 camadas intermediárias, o tempo de treinamento ficou aproximadamente 3 segundos maior do que o menor tempo, 14 segundos para 3 camadas. Como essa diferença de tempo não é significativa e a precisão aumentou, o número de camadas intermediárias adotadas foi 5.

4.5.8 Taxa de aprendizagem

O último ajuste realizado foi o da taxa de aprendizagem. A taxa de aprendizagem ajustada é na verdade um parâmetro do algoritmo otimizador e não propriamente da rede neural artificial. Kingma e Ba (2015), os criadores do otimizador Adam, sugerem o valor de 0,001 para a taxa de aprendizagem, porém, como esse valor foi testado na resolução de outros problemas, resolveu-se fazer uma varredura de forma logarítmica, desse parâmetro, visando determinar qual taxa de aprendizagem resultaria na maior precisão.

Também foi avaliado o desempenho do AMSGrad, alteração do Adam proposta por Reddi, Kale e Kumar (2018). Essa modificação adiciona memória de longo prazo dos gradientes ao otimizador, e, conforme as conclusões do autor, além de evitar problemas de convergência, resultou em aumento de desempenho em alguns casos.

A varredura foi feita duas vezes, uma vez com o AMSGrad habilitado, e outra desabilitado. O resultado das duas varreduras está apresentado na Figura 23. A precisão chegou ao pico de 98,3%, com a taxa de aprendizagem de 0,0005 e o AMSGrad habilitado.

4.5.9 Resumo dos parâmetros

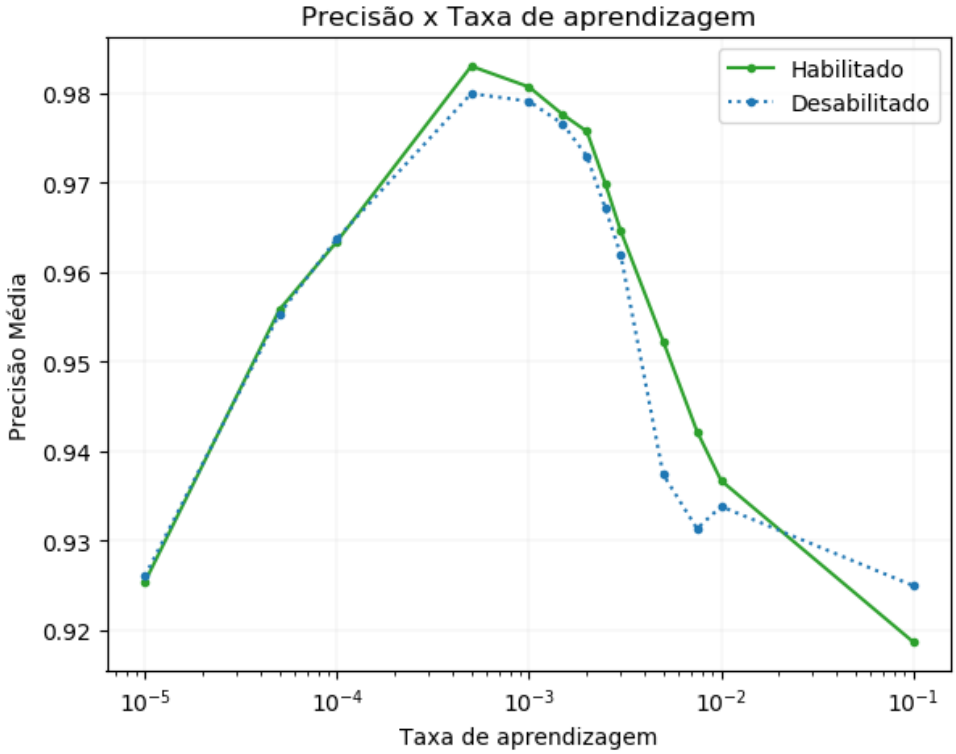
A Tabela 5 apresenta um resumo com os parâmetros adotados após os ajustes.

Tabela 5 - Parâmetros finais do modelo

Parâmetro	**Valor**
Função Custo	*Cross-entropy*
Otimizador	Adam
Função de Ativação	ReLU
Número de Épocas	110
Batch Size	55
Número de neurônios nas camadas intermediárias	25
Número de camadas intermediárias	5
Taxa de aprendizagem	0,0005
AMSGrad	Habilitado

Fonte: Autor.

Figura 23 - Precisão x Taxa de aprendizagem



Fonte: Autor.

5 RESULTADOS

Nesta seção serão apresentados e discutidos os resultados do projeto, levando em consideração os objetivos gerais e específicos definidos no início do projeto e o que foi desenvolvido durante o curso do semestre.

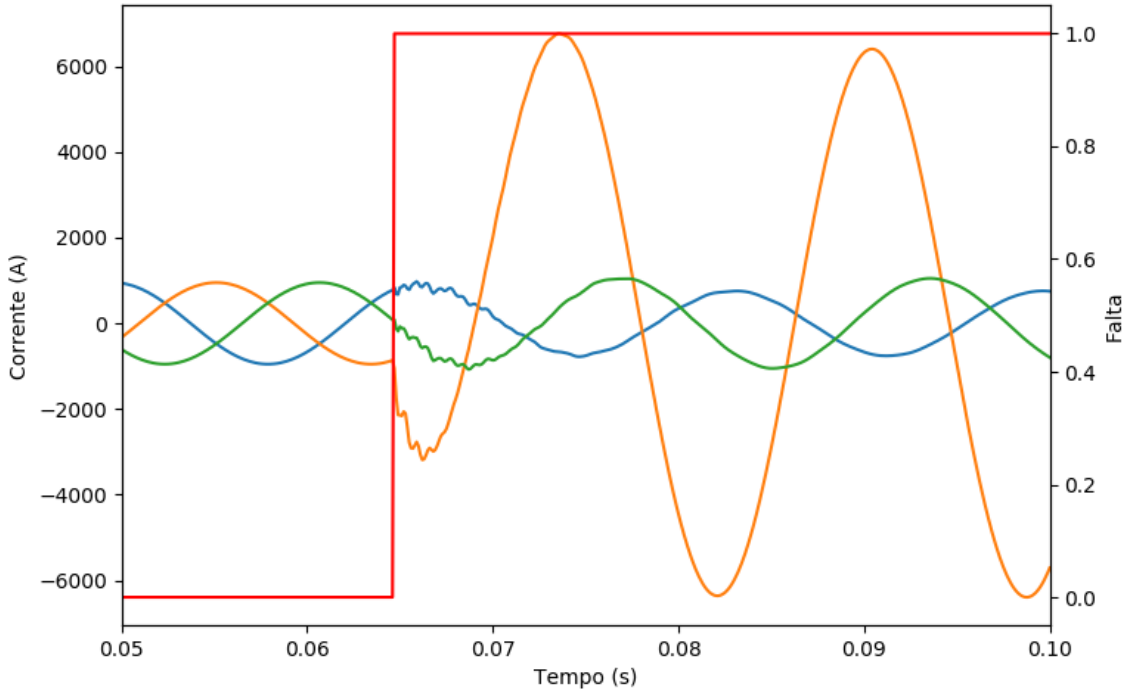
5.1 Detecção e classificação de faltas

O objetivo geral do projeto foi desenvolver um sistema de detecção e classificação de faltas em linhas de transmissão baseado redes neurais artificiais. Esse objetivo foi alcançado através da implementação descrita na seção 4.4.

A Figura 24 apresenta o resultado da rede neural artificial identificando uma falta fase-terra. As linhas em azul, verde e laranja representam a corrente trifásica, entradas da rede neural, assim como a tensão trifásica, e a linha vermelha, apresenta a saída da rede neural que indica falta fase-terra na fase A. Da mesma maneira, a Figura 25 apresenta a rede neural identificando uma falta trifásica.

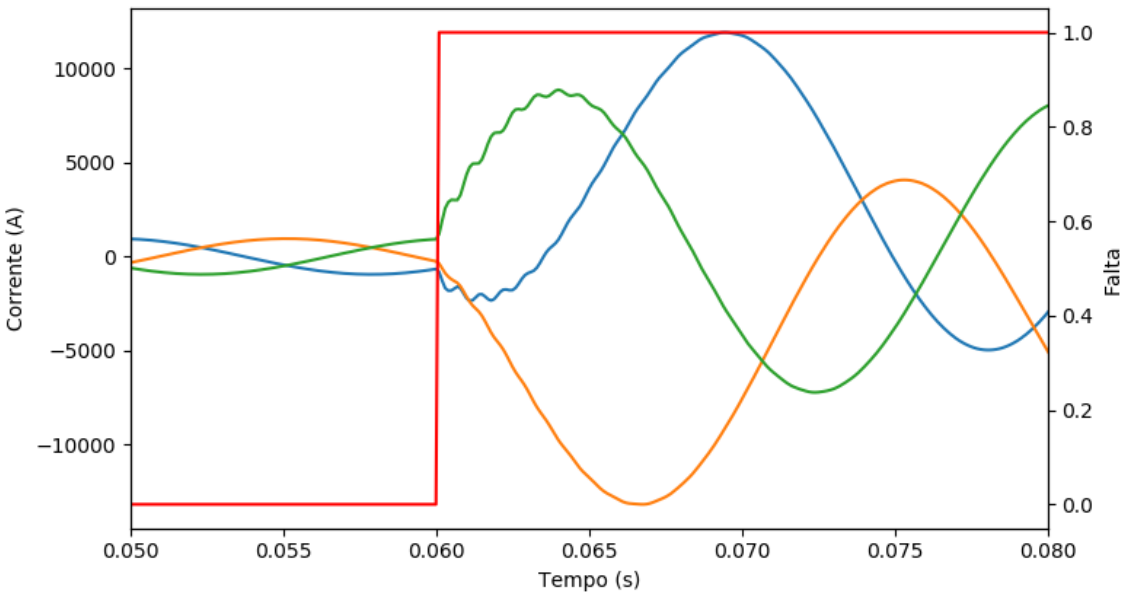
O primeiro objetivo específico do trabalho foi a utilização somente de tensão e corrente trifásica para o desenvolvimento da rede neural artificial. Essas duas informações foram as únicas utilizadas, totalizando em 6 neurônios de entrada, e o modelo conseguiu convergir de forma satisfatória.

Figura 24 - Detecção de falta fase-terra



Fonte: Autor.

Figura 25 - Detecção de falta fase-fase-fase



Fonte: Autor

Ainda relacionado ao desenvolvimento da rede neural, o segundo objetivo específico referiu-se à utilização de tecnologias que facilitassem o desenvolvimento, a experimentação e

a manutenção do *software*. Visando atingir esses requisitos, foram utilizadas a linguagem de programação Python e as bibliotecas Keras, Tensorflow e Scikit-learn. A quantidade e a abrangência dos ajustes apresentados na seção 4.5 demonstraram que esse objetivo foi atingido.

5.2 Precisão final do modelo

A partir da rede neural artificial implementada e dos parâmetros determinados nos ajustes realizados, foi avaliada a precisão final do modelo proposto. O processo do cálculo da precisão foi realizado com validação cruzada e *k-fold* com fator 20. Foram utilizados todas os exemplos existentes no banco de dados, totalizando 549549 amostras de operação da rede. O resultado está apresentado na Tabela 6.

Tabela 6 - Precisão final da rede neural artificial

Número de amostras	**Precisão**	**Desvio Padrão**
549549	99,83%	0,0172%

Fonte: Autor

O resultado é muito satisfatório, e mostra que a rede neural artificial desenvolvida alcançou uma alta taxa de acerto para o conjunto de dados gerados. A taxa de acerto superou a estabelecida no terceiro objetivo específico, que era de 95%.

Cabe ressaltar aqui que, por mais que o modelo proposto tenha alcançado uma alta precisão, ele não necessariamente está pronto para entrar em operação em um sistema de proteção. A modelagem dos casos de falta não levou em consideração fatores como ruídos, harmônicas, saída de outras linhas do sistema ou sobrecargas. A adição desses fatores poderia influenciar na precisão final, podendo até ser necessário realizar novos ajustes nos parâmetros da rede neural.

5.3 Base de dados de faltas

Desde a concepção do projeto já se sabia da necessidade de uma base de dados de faltas para o treinamento e validação da rede neural. A partir desse problema, foi posto como um dos objetivos específicos criar uma base de dados que possa ser utilizada em outros projetos de detecção e classificação de faltas, não necessariamente baseados em redes neurais.

Todo o processo de desenvolvimento e o resultado obtido na construção da base de dados foram descritos no item 384.3 e demonstram que o objetivo específico foi alcançado. A base de dados possui 549 casos de operação, totalizando 549549 amostras e foi disponibilizada de duas formas: uma no SGBD Cassandra e outra em um arquivo único em um formato de fácil importação em ambiente Python.

5.4 Comparação com outro modelo proposto

Visto que o sistema elétrico e os parâmetros utilizados para a criação da base de dados foram baseados no trabalho de Oliveira (2005), julgou-se necessário realizar uma comparação dos resultados obtidos pelo autor e os resultados do presente trabalho. A Figura 26 apresenta a precisão final do modelo proposto pelo autor para o sistema de testes utilizado.

Figura 26 - Resultados de Oliveira (2005)

CONJUNTO	NÚMERO DE AMOSTRAS	ACERTOS
Validação	7718	99.91%
Teste	7717	99.95%
Extra	4410	99.9%

Fonte: Oliveira, 2005, pág. 83.

Levando em consideração o percentual de acertos obtidos no conjunto Extra, a precisão final encontrada pelo autor ficou 0,07% maior do que a encontrada no presente trabalho. Por mais que a diferença seja pequena, ela não deve ser desconsiderava, pois equipamentos de proteção de sistemas elétricos de potência requerem uma grande confiabilidade na sua operação, já que qualquer falha pode acarretar em um grande custo.

Duas importantes diferenças entre os dois trabalhos são a utilização de um filtro *anti-aliasing* para o pré-processamento dos dados e a utilização de valores em por unidade (*p.u.*) como entrada da rede neural. Essas duas técnicas utilizadas por Oliveira (2005) auxiliam no desempenho da rede neural, aumentando a sua precisão final e deixando ela menos sensível a ruídos e contingências no sistema, podendo ser esse o motivo da precisão encontrada pelo autor ser maior do que a encontrada no presente trabalho.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver uma rede neural artificial capaz de detectar e classificar faltas em linhas de transmissão de energia elétrica. O projeto foi elaborado através de elementos utilizados nas pesquisas de Wong, Ryan e Tindle (1996), Oleskovicz (2001), Brahma e Girgis (2004), Oliveira (2005), Moreto (2005), Prasad e Edward (2017) e Farias (2017).

Como base teórica foram revisadas as duas áreas inerentes ao projeto: faltas em linhas de transmissão e redes neurais artificiais. Sobre as faltas, foram descritos cada um dos tipos de curto-circuito, junto com as suas principais causas. A respeito das redes neurais, foi explicado o seu conceito e foram apresentados os neurônios artificiais e biológicos. Foram descritas as funções de ativações mais utilizadas, assim com as arquiteturas de rede. Por fim, foi minuciado o algoritmo *backpropagation*, utilizado para o treinamento de redes neurais.

Após a revisão bibliográfica foi apresentado o desenvolvimento do projeto, detalhando as etapas de modelagem do sistema elétrico, criação da base de dado de faltas, implementação da rede neural artificial e os ajustes dos seus parâmetros.

Os resultados obtidos foram satisfatórios, sendo alcançado o objetivo geral e todos os específicos dentro do cronograma de andamento.

6.1 Sugestões de desenvolvimentos futuros

Seguindo o tema do trabalho e tendo em vista os resultados obtidos, os seguintes pontos são pertinentes para propostas de trabalhos futuros:

- Avaliar o desempenho de redes neurais recorrentes. Esse outro tipo de rede neural artificial é caracterizado por possuir memória de curto prazo e é especialmente útil para a análise de séries temporais. Devido a característica do problema de detecção de faltas, as redes neurais recorrentes se apresentam como um candidato promissor;

- Adicionar fatores como ruídos, harmônicas, saída de outras linhas do sistema ou sobrecargas nas simulações da base de dados de falta. A adição desses fatores gera uma maior complexidade para a base de dados e aumentam a confiabilidade de sistemas treinados e validados por ela;

- Aplicar técnicas de pré-processamentos dos dados de entrada na rede neural. O pré-processamento dos dados tende a melhorar o desempenho do sistema, conforme pode ser visto na comparação realizada com o trabalho de Oliveira (2005).

REFERÊNCIAS

ANACONDA; **What is Anaconda? – Anaconda.** Disponível em: <https://www.anaconda.com/what-is-anaconda/>. Acesso em 05 de junho de 2018.

BAQUI, Ibrahem; ZAMORA, Inmaculada; MAZÓN, Javier; BUIGUES, Garikoitz. High impedance fault detection using wavelet transform and artificial neural networks. In: **Electric Power Systems Research**. (2011), 1325-1333.

BERGEN, Arthur R.; VITTAL, Vijay. **Power systems analysis**. – 2. ed. – New Jersey: Prentice-Hall, 2000.

BRAGA, Antônio P.; CARVALHO, André P. L. F. de; LUDERMIR, Teresa B. **Redes Neurais Artificiais: Teoria e Aplicações**. – 2. ed. – Rio de Janeiro: LTC, 2011.

BRAHMA, Sukumar M; GIRGIS, Adly A. Fault Location on a Transmission Line Using Synchronized Voltage Measurements. In: **IEEE TRANSACTIONS ON POWER DELIVERY**, 19, 2004.

CASSANDRA; **Apache Cassandra**. Disponível em: <http://cassandra.apache.org/> Acesso em 05 de junho de 2018.

CHOLLET, François; **Keras**, 2015 Disponível em: <https://keras.io> Acesso em 05 de junho de 2018.

DAS, Debapriya; **Electrical Power Systems**. 1 ed. New Delhi: New Age International, 2006.

DOCKER. **What is Docker?** Disponível em: <https://www.docker.com/what-docker> Acesso em 5 de junho de 2018.

EMTP-ATP; **World-wide mostly used transients program**. Disponível em: <http://emtp-atp.de/> Acesso em 5 de junho de 2018.

FARIAS, Patrick Escalante. **Rede Neural Continuamente Treinada para Localização de Faltas de Alta Impedância em Redes de Distribuição de Energia Elétrica**, Tese de Doutorado, UFSM, 2017.

FAULKENBERRY, Luces M.; COFFER, Walter. **Electrical power distribution and transmission**. Columbus: Prentice Hall, 1996.

GHADERI, Amin; GINN, Herbert L.; MOHAMMADPOUR, Hossein A.; High impedance fault detection: A review. In: **Electric Power Systems Research**. (2017), 376 – 388.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. – 1.ed. – MIT Press, 2016.

GRAUPE, Daniel. **Principles of Artificial Neural Networks**. Singapore: World Scientific Publishing, 1997.

HAQUE, M. Tarafdar; KASHTIBAN, A. M. Application of Neural Networks in Power Systems; A Review. In: **World Academy of Science, Engineering and Technology**, 6, 2005.

HAYKIN, Simon. **Redes Neurais: Princípios e prática**. – 2. ed. – São Paulo: Prentice-Hall, 2001.

KALOGIROU, Soteris A. **Applied Energy**: Applications of artificial neural-networks for energy systems. Elsevier, 2000.

KINGMA, Diederik P.; BA, Jimmy Lei. Adam: A Method for Stochastic Optimization. In: **International Conference on Learning Representations**, 3, 2015.

MADAN, S.; BOLLINGER, K. E. Application of artificial intelligence in power systems. In: **Electric Power Systems Research**, 41, (1997), 117-131.

MEHROTRA, Kishan; MOHAN, Chilukuri K.; RANKA, Sanjay. **Elements of Artificial Neural Networks**. MIT Press, 1996.

MORETO, Miguel. **Localização de Faltas de Alta Impedância em Sistemas de Distribuição de Energia**: Uma Metodologia Baseada em Redes Neurais Artificiais. Porto Alegre, 2005. 125p. Escola de Engenharia, Universidade Federal do Rio Grande do Sul.

MYSQL; **MySQL Workbench**. Disponível em: <https://www.mysql.com/products/workbench/>. Acesso em 05 de junho de 2018.

NAIR, Vinod; HINTON, Geoffrey E.; Rectified Linear Units Improve Restricted Boltzmann Machines. In: **International Conference of Machine Learning**, 27, 2010.

NIELSEN, Michael A. **Neural networks and Deep Learning**. Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/>. Acesso em 9 de junho de 2018.

OLESKOVICZ, M. **Aplicação de redes neurais artificiais na proteção de distância**. São Carlos, 2001. 215p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

OLIVEIRA, Ângelo Rocha. **Redes Neurais Artificiais Aplicadas na Detecção, Classificação e Localização de Defeitos em Linhas de Transmissão**, Dissertação de Mestrado, UFJF, 2005

ORACLE; **About the Java Technology**. Disponível em: <https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>. Acesso em 05 de junho de 2018.

PEDREGOSA, F. *et al.* Scikit-learn: Machine Learning in Python. In: **Journal of Machine Learning Research**, 12, (2011), 2825-2830.

PINTO, Milton de Oliveira. **Energia elétrica: geração, transmissão e sistemas interligados**. 1 ed. Rio de Janeiro: LTC, 2014.

PRASAD, Avagaddi; EDWARD, Belwin J. Importance of Artificial Neural Networks for Location of Faults in Transmission Systems: A Survey. In: **International Conference on Intelligent Systems and Control,** 11, (2017), 357-362.

PYTHON; **General Python FAQ – Python 3.6 documentation**. Disponível em: <https://docs.python.org/3/faq/general.html>. Acesso em 5 de junho de 2018.

REDDI, Sashank J.; KALE, Satyen; KUMAR, Sanjiv. On the convergence of Adam and Beyond. In: **International Conference on Learning Representations**, 6, 2018.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence**: A Modern Approach. New Jersey: Prentice-Hall, 1995.

TENSORFLOW; **An open-source software library for Machine Intelligence**. Disponível em: <https://www.tensorflow.org/>. Acesso em: 5 de junho de 2018.

VANKAYALA, Vidya; RAO, Nutakki. **Electric Power Systems Research**: Artificial neural networks and their applications to power systems - a bibliographical survey. Elsevier, 1993.

WONG, K. C. P.; RYAN, H. M.; TINDLE, J.; Power system fault prediction using artificial neural networks. In: **Progress in Neural Information Processing**. (1996), 1181 – 1186.

ZANETTA Júnior, Luiz Cera. **Fundamentos de sistemas elétricos de potência** – 1. ed. – São Paulo: Editora Livraria da Física, 2005.

APÊNDICE A – ARQUIVO DE ENTRADA DO ATP

```
BEGIN NEW DATA CASE
C  dT  >< Tmax >< Xopt >< Copt >
  0.0001      .1
     500       1       1       1       1                       1
C        1         2         3         4         5         6         7         8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
C < n1 >< n2 ><ref1><ref2>< R  >< L  >< C  >
$VINTAGE, 1
C Impedancia da falta
  FPTA                                10.0 0.000000000E+00 0.000000000E+00     0
  FPTB                                10.0 0.000000000E+00 0.000000000E+00     0
  FPTC                                10.0 0.000000000E+00 0.000000000E+00     0
C Linha antes da falta
-1P1A   166A              5.08621E-01 2.02913E+00 8.33303E-03       100.0 0 00
-2P1B   166B              4.69600E-02 1.05244+00  1.12962E-02       100.0 0 00
-3P1C   166C
C Linha apos da falta
-1166A  T0A               5.08621E-01 2.02913E+00 8.33303E-03        50.0 0 00
-2166B  T0B               4.69600E-02 1.05244+00  1.12962E-02        50.0 0 00
-3166C  T0C
C Linha TR
-1T2A   RA                5.08621E-01 2.02913E+00 8.33303E-03 1.00000E+02 0 00
-2T2B   RB                4.69600E-02 1.05244+00  1.12962E-02 1.00000E+02 0 00
-3T2C   RC
C Linha TQ
-1T1A   QA                5.08621E-01 2.02913E+00 8.33303E-03 8.00000E+01 0 00
-2T1B   QB                4.69600E-02  1.05244+00 1.12962E-02 8.00000E+01 0 00
-3T1C   QC
$VINTAGE, 0
51PA    GERPA            .31820 9.62069E+00
52PB    GERPB            1.4035 4.24403E+01
53PC    GERPC
51Q1A   GERQA            .35350 1.06896E+01
52Q1B   GERQB            1.5595 4.71560E+01
53Q1C   GERQC
51R1A   GERRA            .33490 1.01273E+01
52R1B   GERRB            1.4774 4.46737E+01
53R1C   GERRC
$VINTAGE, 1
$VINTAGE, 0
BLANK card terminating network
C < n 1>< n 2>< Tclose ><Top/Tde ><   Ie   ><Vf/CLOP ><  type  >
C Chaves para as faltas
  166A  FPTA          1. 1.000E+02 0.000E+00                                  0
  166B  FPTB          1. 1.000E+02 0.000E+00                                  0
  166C  FPTC     0.02957 1.000E+02 0.000E+00                                  0
  166A  166B          1. 1.000E+02 0.000E+00                                  0
  166B  166C          1. 1.000E+02 0.000E+00                                  0
  166C  166A          1. 1.000E+02 0.000E+00                                  0
C Medicao
  PA    P1A                                         MEASURING               1
  PB    P1B                                         MEASURING               1
  PC    P1C                                         MEASURING               1
C Medicao, desabilitados
  TA    T1A                                         MEASURING               0
  TB    T1B                                         MEASURING               0
  TC    T1C                                         MEASURING               0
  QA    Q1A                                         MEASURING               0
  QB    Q1B                                         MEASURING               0
  QC    Q1C                                         MEASURING               0
  T0A   TA                                          MEASURING               0
  T0B   TB                                          MEASURING               0
```

```
63   T0C   TC                                                MEASURING               0
64   TA    T2A                                               MEASURING               0
65   TB    T2B                                               MEASURING               0
66   TC    T2C                                               MEASURING               0
67   RA    R1A                                               MEASURING               0
68   RB    R1B                                               MEASURING               0
69   RC    R1C                                               MEASURING               0
70 BLANK card terminating switches
71 C < n 1><>< Ampl.  >< Freq.  ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP  >
72 14GERQA  0 3.590E+05 6.000E+01 0.000E+00         0          -1.000E+00 1.000E+02
73 14GERQB  0 3.590E+05 6.000E+01-1.200E+02         0          -1.000E+00 1.000E+02
74 14GERQC  0 3.590E+05 6.000E+01 1.200E+02         0          -1.000E+00 1.000E+02
75 14GERPA  0 3.593E+05 6.000E+01 2.000E+01         0          -1.000E+00 1.000E+02
76 14GERPB  0 3.593E+05 6.000E+01-1.000E+02         0          -1.000E+00 1.000E+02
77 14GERPC  0 3.593E+05 6.000E+01 1.400E+02         0          -1.000E+00 1.000E+02
78 14GERRA  0 2.930E+05 6.000E+01 1.000E+01         0          -1.000E+00 1.000E+02
79 14GERRB  0 2.930E+05 6.000E+01-1.100E+02         0          -1.000E+00 1.000E+02
80 14GERRC  0 2.930E+05 6.000E+01 1.300E+02         0          -1.000E+00 1.000E+02
81 BLANK card terminating sources
82   PA    PB    PC
83 BLANK card terminating outputs
84 BLANK card terminating plots
85 BEGIN NEW DATA CASE
```

UNIVATES

R. Avelino Talini, 171 | Bairro Universitário | Lajeado | RS | Brasil
CEP 95914.014 | Cx. Postal 155 | Fone: (51) 3714.7000
www.univates.br | 0800 7 07 08 09